

FFFFFFF FFFF	DDDDDDDDDDDD	LLL
FFFFFFF FFFF	DDDDDDDDDDDD	LLL
FFFFFFF FFFF	DDDDDDDDDDDD	LLL
FFF	DDD	DDD
FFF	DDD	LLL
FFF	DDDDDDDDDDDD	LLLLLLLLLLLL
FFF	DDDDDDDDDDDD	LLLLLLLLLLLL
FFF	DDDDDDDDDDDD	LLLLLLLLLLLL

\*\*FILE\*\*ID\*\*FDLPARSE

85

FDL  
V04

FF FFFFFFFF FF DDDDDDDD LL PPPPPPPP AA RRRRRRRR SS EEEEEEEE  
FF FFFFFFFF FF DDDDDDDD LL PPPPPPPP AA RRRRRRRR SS EEEEEEEE  
FF DD DD LL PP PP AA AA RR RR SS EE  
FF DD DD LL PP PP AA AA RR RR SS EE  
FF DD DD LL PP PP AA AA RR RR SS EE  
FF DD DD LL PP PP AA AA RR RR SS EE  
FF FFFFFF DD DD LL PPPPPPPP AA AA RRRRRRRR SS EEEEEEEE  
FF FFFFFF DD DD LL PPPPPPPP AA AA RRRRRRRR SS EEEEEEEE  
FF DD DD LL PP AAAA RR RR SS EE  
FF DD DD LL PP AAAA RR RR SS EE  
FF DD DD LL PP AA RR RR SS EE  
FF DD DD LL PP AA RR RR SS EE  
FF DDDDDDDD LLLLLLLL PP AA RR RR SS EEEEEEEE  
FF DDDDDDDD LLLLLLLL PP AA RR RR SS EEEEEEEE

```
1 0001 0 %TITLE 'FDL$PARSE'
2 0002 0 %SBTTL 'FDL Parse Action Routines'
3 0003 0 MODULE FDLPARSE      ( IDENT='V04-000',
4                                ADDRESSING_MODE ( EXTERNAL = GENERAL ),
5                                ADDRESSING_MODE ( NONEXTERNAL = GENERAL ),
6                                OPTLEVEL=3
7                                ) =
8
9 0009 1 BEGIN
10 0010 1
11 0011 1 *****
12 0012 1 */
13 0013 1 */ COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
14 0014 1 */ DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
15 0015 1 */ ALL RIGHTS RESERVED.
16 0016 1 */
17 0017 1 */ THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
18 0018 1 */ ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
19 0019 1 */ INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
20 0020 1 */ COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
21 0021 1 */ OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
22 0022 1 */ TRANSFERRED.
23 0023 1 */
24 0024 1 */ THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
25 0025 1 */ AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
26 0026 1 */ CORPORATION.
27 0027 1 */
28 0028 1 */ DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
29 0029 1 */ SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
30 0030 1 */
31 0031 1 */
32 0032 1 *****/
```

```
34 0033 1 ++  
35 0034 1  
36 0035 1 Facility: RMS-32 FDL Utilities  
37 0036 1  
38 0037 1 Environment: VAX/VMS Operating System  
39 0038 1  
40 0039 1  
41 0040 1  
42 0041 1 Abstract:  
43 0042 1 Routines which fill the rms control blocks  
44 0043 1 for the FDL parser  
45 0044 1  
46 0045 1 Contents:  
47 0046 1 INIT_PARSE  
48 0047 1 LINE_PARSED  
49 0048 1 SET_AREA_P  
50 0049 1 SET_DATE_P  
51 0050 1 SET_JNL_P  
52 0051 1 SET_ACL_P  
53 0052 1 SET_FILE_P  
54 0053 1 SET_KEY_P  
55 0054 1 SET_RECORD_P  
56 0055 1 SET_ACCESS_P  
57 0056 1 SET_SHARING_P  
58 0057 1 SET_CONNECT_P  
59 0058 1 SET PROT  
60 0059 1 ALLOCATE_XAB  
61 0060 1 FIND_ID  
62 0061 1 GET_VM  
63 0062 1 FREE_VM  
64 0063 1  
65 0064 1 --
```

67 0065 1 |  
68 0066 1 | Author: Keith B Thompson Creation date: July-1981  
69 0067 1 |  
70 0068 1 |  
71 0069 1 | Modified by:  
72 0070 1 |  
73 0071 1 | V03-011 RRB0015 Rowland R. Bradley 29 Feb 1984  
74 0072 1 | Comment out references to ERASE\_ON\_DELETE and ACL support.  
75 0073 1 | Not supported for V4.0.  
76 0074 1 |  
77 0075 1 | V03-010 RRB0008 Rowland R. Bradley 19 Jan 1984  
78 0076 1 | Support NULL strings in file name.  
79 0077 1 |  
80 0078 1 | V03-009 KFH0007 Ken Henderson 10 Sep 1983  
81 0079 1 | Support for named UICs  
82 0080 1 |  
83 0081 1 | V03-008 KFH0006 Ken Henderson 29 Jul 1983  
84 0082 1 | Check status of call to LIBS...  
85 0083 1 | Added DEFERRED\_WRITE, ERASE\_ON\_DELETE  
86 0084 1 |  
87 0085 1 | V03-007 KFH0005 Ken Henderson 6 Jan 1983  
88 0086 1 | Fixed allocation of keyname buffer  
89 0087 1 |  
90 0088 1 | V03-006 KFH0004 Ken Henderson 21 Dec 1982  
91 0089 1 | Deleted unused ref to tpa\_block  
92 0090 1 |  
93 0091 1 | V03-005 KFH0003 Ken Henderson 22 Nov 1982  
94 0092 1 | Add support for default and main  
95 0093 1 | parses in FDL\$PARSE  
96 0094 1 | Fix FDL\$SFREE\_VM to signal status  
97 0095 1 |  
98 0096 1 | V03-004 KFH0002 Ken Henderson 6-Oct-1982  
99 0097 1 | Add support for Journal, Access,  
100 0098 1 | ACL, Sharing, Connect primaries  
101 0099 1 |  
102 0100 1 | V03-003 KBT0069 Keith B. Thompson 24-Jun-1982  
103 0101 1 | Initialize the length in fdl\$ab\_item  
104 0102 1 |  
105 0103 1 | V03-002 KBT0030 Keith Thompson 30-Mar-1982  
106 0104 1 | Fix error processing of the date & time stuff  
107 0105 1 |  
108 0106 1 | V03-001 KFH0001 Ken Henderson 29 March 1982  
109 0107 1 | Fixed SET AREA\_P to set LBN  
110 0108 1 | instead of VBN for volume placement  
111 0109 1 |  
112 0110 1 |\*\*\*\*

```
114 0111 1
115 0112 1 PSECT
116 0113 1      OWN      = FDL$OWN      (PIC).
117 0114 1      GLOBAL   = FDL$GLOBAL  (PIC).
118 0115 1      PLIT     = FDL$PLIT    (SHARE,PIC).
119 0116 1      CODE     = FDL$CODE    (SHARE,PIC);
120 0117 1
121 0118 1 LIBRARY 'SYSSLIBRARY:STARLET';
122 0119 1 REQUIRE 'SRC$:FDLUTIL';
123 0304 1 REQUIRE 'LIBS:FDLPARDEF';
124 0843 1
125 0844 1 EXTERNAL ROUTINE
126 0845 1      LIB$GET_VM,
127 0846 1      LIB$FREE_VM,
128 0847 1      FDL$SRMS_ERROR : NOVALUE;
129 0848 1
130 0849 1 DEFINE_ERROR_CODES;
131 0850 1
132 0851 1 FORWARD ROUTINE
133 0852 1      SET_AREA_P : NOVALUE.
134 0853 1      SET_DATE_P : NOVALUE.
135 0854 1      SET_JNL_P : NOVALUE.
136 0855 1      SET_ACL_P : NOVALUE.
137 0856 1      SET_FILE_P : NOVALUE.
138 0857 1      SET_KEY_P : NOVALUE.
139 0858 1      SET_RECORD_P : NOVALUE.
140 0859 1      SET_ACCESS_P : NOVALUE.
141 0860 1      SET_SHARING_P : NOVALUE.
142 0861 1      SET_CONNECT_P : NOVALUE.
143 0862 1      SET_PROT : NOVALUE.
144 0863 1      ALLOCATE_XAB.
145 0864 1      FIND_ID : NOVALUE.
146 0865 1      FDL$GET_VM,
147 0866 1      FDL$FREE_VM : NOVALUE;
148 0867 1
149 0868 1 EXTERNAL
150 0869 1      FDLSAB_TPARSE_BLOCK : BLOCK [,BYTE].
151 0870 1      FDLSAB_ITEM : DESC_BLK.
152 0871 1      FDLSAB_CTRL : BLOCK [,BYTE].
153 0872 1      FDLSGL_PCALL,
154 0873 1      FDLSGL_STMNTPNUM,
155 0874 1      FDLSGL_PRIMARY,
156 0875 1      FDLSGL_PRINUM,
157 0876 1      FDLSAB_PRICTRL,
158 0877 1      FDLSGL_SECONDARY,
159 0878 1      FDLSGL_SECNUM,
160 0879 1      FDLSGL_QUALIFIER,
161 0880 1      FDLSGL_NUMBER,
162 0881 1      FDLSGL_SWITCH,
163 0882 1      FDLSGL_OWNER_UIC,
164 0883 1      FDLSGL_SPARET,
165 0884 1      FDLSGL_PROTECTION,
166 0885 1      FDLSGL_FID1,
167 0886 1      FDLSGL_FID2,
168 0887 1      FDLSGL_FID3,
169 0888 1      FDLSAB_AREA_BKZ : REF VECTOR [,BYTE].
170 0889 1      FDLSAL_DATE_TIME : VECTOR [,LONG].
```

```
171      0890 1     FDL$AB_STRING          : DESC_BLK,  
172      0891 1  
173      0892 1     FDL$AB_PARSED_FAB    : REF $FAB_DECL,  
174      0893 1     FDL$AB_PARSED_RAB    : REF $RAB_DECL;  
175      0894 1  
176      0895 1     LITERAL  
177      0896 1     SPACE   = 32;  
178      0897 1  
179      0898 1     OWN  
180      0899 1     HIGHEST_AREA_NO : BYTE,  
181      0900 1     CURRENT_XAB    : REF BLOCK [ ,BYTE ],  
182      0901 1     END_XAB       : REF BLOCK [ ,BYTE ],  
183      0902 1  
184      0903 1     JNL_XAB        : REF $XABJNL_DECL, | Journal XAB  
185      0904 1     DATE_XAB       : REF $XABDAT_DECL, | Date XAB  
186      0905 1     REVISION_XAB  : REF $XABRDT_DECL, | Revision Date and Time XAB  
187      0906 1     PROTECTION_XAB: REF $XABPRO_DECL; | Protection XAB  
188      0907 1
```

```
190 0908 1 %SBTTL 'INIT_PARSE'  
191 0909 1 GLOBAL ROUTINE FDLS$INIT_PARSE : NOVALUE =  
192 0910 1 !++  
193 0911 1  
194 0912 1 Functional Description:  
195 0913 1  
196 0914 1 Init variables and allocate a buffer for the area bucket sizes  
197 0915 1  
198 0916 1 Calling Sequence:  
199 0917 1  
200 0918 1 fdls$init_parse()  
201 0919 1  
202 0920 1 Input Parameters:  
203 0921 1 none  
204 0922 1  
205 0923 1 Implicit Inputs:  
206 0924 1 none  
207 0925 1  
208 0926 1 Output Parameters:  
209 0927 1 none  
210 0928 1  
211 0929 1 Implicit Outputs:  
212 0930 1 none  
213 0931 1  
214 0932 1 Routine Value:  
215 0933 1 none  
216 0934 1  
217 0935 1 Routines Called:  
218 0936 1  
219 0937 1 lib$get_vm  
220 0938 1  
221 0939 1 Side Effects:  
222 0940 1  
223 0941 1 Allocates a buffer pointed to by FDLSAB_AREA_BKZ  
224 0942 1  
225 0943 1 --  
226 0944 1  
227 0945 2 BEGIN  
228 0946 2  
229 0947 2 LOCAL  
230 0948 2 BYTES:  
231 0949 2  
232 0950 2 ! Set the parse control bits  
233 0951 2  
234 0952 2 FDLSAB_CTRL [ FDLSV_STATUS ] = _SET;  
235 0953 2 FDLSAB_CTRL [ FDLSV_INITIAL ] = _SET;  
236 0954 2  
237 0955 2 ! Clear the other CTRL bits except the following ones:  
238 0956 2 PCALL  
239 0957 2 DCL  
240 0958 2 STRING_SPEC  
241 0959 2 GCALL  
242 0960 2  
243 0961 2 FDLSAB_CTRL [ FDLSV_WARNING ] = _CLEAR;  
244 0962 2 FDLSAB_CTRL [ FDLSV_PRIMARY ] = _CLEAR;  
245 0963 2 FDLSAB_CTRL [ FDLSV_NEWPRI ] = _CLEAR;  
246 0964 2 FDLSAB_CTRL [ FDLSV_SECONDARY ] = _CLEAR;
```

```

247      0965 2    FDLSAB_CTRL [ FDLSV_COMMENT ] = _CLEAR;
248      0966 2    FDLSAB_CTRL [ FDLSV_LINECMT ] = _CLEAR;
249      0967 2    FDLSAB_CTRL [ FDLSV_APOST_PRES ] = _CLEAR;
250      0968 2    FDLSAB_CTRL [ FDLSV_QUOTE_PRES ] = _CLEAR;
251      0969 2    FDLSAB_CTRL [ FDLSV_USED_STRING ] = _CLEAR;
252      0970 2
253      0971 2    ! Initialize the item length for fdl$get_line
254      0972 2
255      0973 2    FDLSAB_ITEM [ DSC$W_LENGTH ] = 0;
256      0974 2
257      0975 2    IF NOT .FDLSAB_CTRL [ FDLSV_REPARSE ]
258      0976 2    THEN
259      0977 2    BEGIN
260      0978 2
261      0979 2    ! Clear the pointers to xabs
262      0980 2
263      0981 2    JNL_XAB      = _CLEAR;
264      0982 2    DATE_XAB     = _CLEAR;
265      0983 2    REVISION_XAB = _CLEAR;
266      0984 2    PROTECTION_XAB = _CLEAR;
267      0985 2
268      0986 2    END:
269      0987 2
270      0988 2    ! Clear misc
271      0989 2
272      0990 2    FDL$GL_STMNTNUM = 0;
273      0991 2    FDLSAB_PRICTRL = _CLEAR;
274      0992 2    CURRENT_XAB   = _CLEAR;
275      0993 2    HIGHEST_AREA_NO = 0;
276      0994 2
277      0995 2    ! Allocate memory for the area bucket size array NOTE: Use lib$get_vm so
278      0996 2    we can return this in fdl$$finish_parse
279      0997 2
280      0998 2    BYTES = 256;
281      0999 2
282      1000 2    IF NOT LIB$GET_VM ( BYTES,FDLSAB_AREA_BKZ )
283      1001 2    THEN
284      1002 2    SIGNAL_STOP ( FDLS_INSVRMEM );
285      1003 2
286      1004 2    ! Zero the values
287      1005 2
288      1006 2    CH$FILL( 0,,BYTES,,FDLSAB_AREA_BKZ );
289      1007 2
290      1008 2    RETURN
291      1009 2
292      1010 1    END:

```

.TITLE FDLPARSE VAX-11 FDL Utilities  
.IDENT \V04-000\

.PSECT \_FDL\$OWN,NOEXE, PIC,2

00000 HIGHEST\_AREA\_NO:  
00001 .BLKB 1  
00004 CURRENT\_XAB:

J 5  
16-Sep-1984 01:50:08 VAX-11 Bliss-32 v4.0-742 Page 8  
14-Sep-1984 12:31:19 DISK\$VMSMASTER:[FDL.SRC]FDLPARSE.B32;1 (5)

FDL  
V04

00008	END_XAB:	.BLKB	4
0000C	JNL_XAB:	.BLKB	4
00010	DATE_XAB:	.BLKB	4
00014	REVISION_XAB:	.BLKB	4
00018	PROTECTION_XAB:	.BLKB	4
		.EXTRN	LIB\$GET_VM, LIB\$FREE_VM
		.EXTRN	FDL\$SRMS_ERROR, FDLS_FACILITY
		.EXTRN	FDLS_FAO_MAX, FDLS_ABKW
		.EXTRN	FDLS_ABPRIKW, FDLS_CREATE
		.EXTRN	FDLS_CREATED, FDLS_CREATEDSTM
		.EXTRN	FDLS_FDLERROR, FDLS_ILL_ARG
		.EXTRN	FDLS_INSVIRMEM, FDLS_INVBLK
		.EXTRN	FDLS_INVDATIM, FDLS_MULPRI
		.EXTRN	FDLS_MULSEC, FDLS_NOQUAL
		.EXTRN	FDLS_NULLPRI, FDLS_OPENFDL
		.EXTRN	FDLS_OUTORDER, FDLS_OPENOUT
		.EXTRN	FDLS_WRITEERR, FDLS_READERR
		.EXTRN	FDLS_RFLOC, FDLS_TITLE
		.EXTRN	FDLS_SYNTAX, FDLS_VALPRI
		.EXTRN	FDLS_UNQUAKW, FDLS_UNPRIKW
		.EXTRN	FDLS_UNSECKW, FDLS_WARNING
		.EXTRN	FDLSAB_TPARSE_BLOCK
		.EXTRN	FDLSAB_ITEM, FDLSAB_CTRL
		.EXTRN	FDL\$GL_PCALL, FDLSGE_STMNTNUM
		.EXTRN	FDL\$GL_PRIMARY, FDLSGL_PRINUM
		.EXTRN	FDL\$AB_PRICTRL, FDLSGL_SECONDARY
		.EXTRN	FDL\$GL_SECNUM, FDLSGL_QUALIFIER
		.EXTRN	FDL\$GL_NUMBER, FDLSGL_SWITCH
		.EXTRN	FDL\$GL_OWNER_UIC
		.EXTRN	FDL\$GL_SPARET, FDLSGL_PROTECTION
		.EXTRN	FDL\$GL_FID1, FDLSGL_FID2
		.EXTRN	FDL\$GL_FID3, FDLSAB_AREA_BKZ
		.EXTRN	FDLSAL_DATE_TIME
		.EXTRN	FDLSAB_STRING, FDLSAB_PARSED_FAB
		.EXTRN	FDLSAB_PARSED_RAB

.PSECT	_FDL\$CODE,NOWRT, SHR, PIC.2	
.ENTRY	FDL\$INIT_PARSE, Save R2,R3,R4,R5,R6,R7,R8	: 0909
MOVAB	FDL\$AB_AREA_BKZ, R8	.
MOVAB	FDL\$AB_CTRL, R7	.
MOVAB	JNL_XAB, R6	.
SUBL2	#4, SP	.
INSV	#1, #0, #3, FDL\$AB_CTRL	0952
BISB2	#128, FDL\$AB_CTRL	0953
BICW2	#58232, FDL\$AB_CTRL	0969
CLRW	FDL\$AB_ITEM	0973
BLBS	FDL\$AB_CTRL+2, 1\$	0975
CLRQ	JNL_XAB	0981
CLRQ	REVISION_XAB	0983
CLRL	FDL\$GL_STMNTNUM	0990
CLRL	FDL\$AB_PRICTRL	0991

FDLPARSE  
V04-000

VAX-11 FDL Utilities  
INIT\_PARSE

K 5  
16-Sep-1984 01:50:08  
14-Sep-1984 12:31:19

VAX-11 BLfss-32 V4.0-742  
DISKSVMMASTER:[FDL.SRC]FDLPARSE.B32;1

Page 9  
(5)

		F8	A6	D4	00043	CLRL	CURRENT_XAB	: 0992
		F4	A6	94	00046	CLRB	HIGHEST_AREA_NO	: 0993
	6E	0100	8F	3C	00049	MOVZWL	#256, BYTES	: 0998
			58	DD	0004E	PUSHL	R8	: 1000
			04	AE	00050	PUSHAB	BYTES	: :
	00000000G	00	02	FB	00053	CALLS	#2. LIB\$GET_VM	: :
		0D	50	E8	0005A	BLBS	R0, 2\$	: :
	00000000G	00	8F	DD	0005D	PUSHL	#FDL\$INSVIRMEM	: 1002
		50	01	FB	00063	CALLS	#1. LIB\$STOP	: :
6E	00	6E	68	DD	0006A	MOVL	FDL\$AB AREA BKZ, R0	: 1006
			00	2C	0006D	MOVCS	#0, (SP), #0, BYTES, (R0)	: :
			60	00	00072			: 1010
			04	00073		RET		

: Routine Size: 116 bytes. Routine Base: \_FDL\$CODE + 0000

FDL  
V04

```
294 1011 1 XSBTTL 'FINISH_PARSE'  
295 1012 1 GLOBAL ROUTINE "FDL$FINISH_PARSE =  
296 1013 1 !++  
297 1014 1  
298 1015 1 Functional Description:  
299 1016 1  
300 1017 1 Ties up any loose ends and returns with the final status value  
301 1018 1  
302 1019 1 Calling Sequence:  
303 1020 1  
304 1021 1     status = fdl$finish_parse()  
305 1022 1  
306 1023 1 Input Parameters:  
307 1024 1  
308 1025 1     none  
309 1026 1  
310 1027 1 Implicit Inputs:  
311 1028 1  
312 1029 1     none  
313 1030 1  
314 1031 1 Output Parameters:  
315 1032 1  
316 1033 1     none  
317 1034 1  
318 1035 1 Implicit Outputs:  
319 1036 1  
320 1037 1     none  
321 1038 1  
322 1039 1 Routine Value:  
323 1040 1  
324 1041 1     SSS_NORMAL      - If everything completed correctly  
325 1042 1     FDLS_WARNING    - If there were warnings during processing  
326 1043 1     FDLS_FDLERROR  - If there were real problems  
327 1044 1  
328 1045 1 Routines Called:  
329 1046 1  
330 1047 1     lib$free_vm  
331 1048 1  
332 1049 1 Side Effects:  
333 1050 1     none  
334 1051 1  
335 1052 1 !--  
336 1053 1  
337 1054 2 BEGIN  
338 1055 2  
339 1056 2 LOCAL  
340 1057 2     STATUS,  
341 1058 2     XAB : REF BLOCK [ ,BYTE ].  
342 1059 2     BYTES;  
343 1060 2  
344 1061 2     ! If successful then continue and return ok  
345 1062 2  
346 1063 2 IF .FDLSAB_CTRL [ FDLSV_STATUS ]  
347 1064 2 THEN  
348 1065 2     STATUS = SSS_NORMAL  
349 1066 2 ELSE  
350 1067 2
```

351 1068 2 | If the problem was a warning then continue and return fdls\_warning  
352 1069 2 | else return imeditaly  
353 1070 2  
354 1071 2 IF .FDLSAB\_CTRL [ FDLSV\_STATUS ] EQLU STSSK\_WARNING  
355 1072 2 THEN STATUS = FDLS\_WARNING  
356 1073 2 ELSE RETURN FDLS\_FDLERROR;  
357 1074 2  
358 1075 2  
359 1076 2  
360 1077 2 | Travel through the xabs and fix up random things  
361 1078 2 | UNLESS THIS IS JUST A DEFAULT PARSE  
362 1079 2  
363 1080 3  
364 1081 4 IF ( NOT .FDLSAB\_CTRL [ FDLSV\_DFLT\_PRES ] )  
365 1082 3 OR ( .FDLSAB\_CTRL [ FDLSV\_REPARSE ] )  
366 1083 4 ) THEN BEGIN  
367 1084 2  
368 1085 3  
369 1086 3  
370 1087 3 XAB = .FDLSAB\_PARSED\_FAB [ FABSL\_XAB ];  
371 1088 3  
372 1089 3 WHILE .XAB NEQU 0  
373 1090 3 DO BEGIN  
374 1091 4  
375 1092 4  
376 1093 4 | If this is a key xab fix the fill factors if neccary  
377 1094 4  
378 1095 4 IF .XAB [ XABSB\_COD ] EQLU XABSC\_KEY  
379 1096 4 THEN BEGIN  
380 1097 5  
381 1098 5  
382 1099 5 | Make sure the area numbers are valid if not simply exit  
383 1100 5 | RMS will catch it during the create  
384 1101 5  
385 1102 5 IF ( .XAB [ XABSB\_DAN ] GTRU .HIGHEST\_AREA\_NO ) OR  
386 1103 6 ( .XAB [ XABSB\_IAN ] GTRU .HIGHEST\_AREA\_NO )  
387 1104 5 THEN EXITLOOP;  
388 1105 5  
389 1106 5  
390 1107 5  
391 1108 5 | Data level fill  
392 1109 6 XAB [ XABSW\_DFL ] = ( .FDLSAB\_AREA\_BKZ [ .XAB [ XABSB\_DAN ] ] \* BLOCK\_SIZE \*  
393 1110 5 .XAB [ XABSW\_DFL ] ) / 100;  
394 1111 5  
395 1112 5 | Index level fill  
396 1113 5  
397 1114 6 XAB [ XABSW\_IFL ] = ( .FDLSAB\_AREA\_BKZ [ .XAB [ XABSB\_IAN ] ] \* BLOCK\_SIZE \*  
398 1115 5 .XAB [ XABSW\_IFL ] ) / 100  
399 1116 4 END;  
400 1117 4  
401 1118 4 XAB = .XAB [ XABSL\_NXT ]  
402 1119 4  
403 1120 4 END;  
404 1121 4  
405 1122 4  
406 1123 4  
407 1124 2 END;  
| Deallocate memory for the area bucket size array

```

408   1125 2      !
409   1126 2      BYTE = 256:
410   1127 3      BEGIN
411   1128 3      LOCAL STATUS:
412   1129 3
413   1130 4      IF NOT ( STATUS = LIB$FREE_VM ( BYTES,FDLSAB_AREA_BKZ ) )
414   1131 3      THEN
415   1132 3      SIGNAL_STOP ( .STATUS );
416   1133 2      END;
417   1134 2
418   1135 2      RETURN .STATUS
419   1136 2
420   1137 1      END;

```

				007C 00000	.ENTRY	FDL\$FINISH_PARSE, Save R2,R3,R4,R5,R6	: 1012
50	65			56 00000000G 00 9E 00002	MOVAB	FDL\$AB_AREA_BKZ, R6	
				55 00000000G 00 9E 00009	MOVAB	FDL\$AB_CTRL, R5	
				5E 04 C2 00010	SUBL2	#4, SP	
				03 00 EF 00013	EXTZV	#0, #3, FDLSAB_CTRL, R0	
				05 50 E9 00018	BLBC	R0, 1\$	: 1063
				53 01 D0 0001B	MOVL	#1, STATUS	: 1065
				53 13 11 0001E	BRB	3\$	
				09 09 12 00020 1\$:	BNEQ	2\$	: 1071
				53 8F D0 00022	MOVL	#FDLS_WARNING, STATUS	: 1073
				50 08 11 00029	BRB	3\$	
				50 8F D0 0002B 2\$:	MOVL	#FDLS_FDLERROR, R0	: 1075
				50 04 00032	RET		
04	02	A5	01	E1 00033 3\$:	BBC	#1, FDLSAB_CTRL+2, 4\$	: 1081
		6B	A5	E9 00038	BLBC	FDL\$AB_CTRL+2, 7\$	: 1083
		02	50	00000000G 00 00 003C 4\$:	MOVL	FDL\$AB_PARSED_FAB, R0	: 1087
		24	50	D0 00043	MOVL	36(R0), XAB	
			5E	13 00047 5\$:	BEQL	7\$	: 1089
		15	60	91 00049	CMPB	(XAB), #21	: 1095
			53	12 0004C	BNEQ	6\$	
		52	A0	9A 0004E	MOVZBL	10(XAB), R2	
		0A	51	00000000 00 9A 00052	MOVZBL	HIGHEST_AREA_NO, R1	: 1102
		51	51	J1 00059	CMPL	R2, R1	
			49	1A 0005C	BGTRU	7\$	
		51	08	A0 91 0005E	CMPB	8(XAB), R1	: 1103
			43	1A 00062	BGTRU	7\$	
		51	51	66 D0 00066	MOVL	FDL\$AB_AREA_BKZ, R1	: 1109
			52	6241 9A 00067	MOVZBL	(R2)[RT], R2	: 1110
			54	1C A0 3C 0006B	MOVZWL	28(XAB), R4	
			52	54 C4 0006F	MULL2	R4, R2	
52	54		52	09 78 00072	ASHL	#9, R2, R2	: 1109
			52	8F C7 00076	DIVL3	#100, R2, R4	: 1110
			A0	54 B0 0007E	MOVW	R4, 28(XAB)	
			52	08 A0 9A 00082	MOVZBL	8(XAB), R2	: 1114
			51	6241 9A 00086	MOVZBL	(R2)[R1], R1	: 1115
			54	1A A0 3C 0008A	MOVZWL	26(XAB), R4	
			51	54 C4 0008E	MULL2	R4, R1	
		51	51	09 78 00091	ASHL	#9, R1, R1	: 1114
		52	51	8F C7 00095	DIVL3	#100, R1, R2	: 1115

FDLPARSE  
V04-000

VAX-11 FDL Utilities  
FINISH\_PARSE

B 6  
16-Sep-1984 01:50:08  
14-Sep-1984 12:31:19

VAX-11 Blfss-32 V4.0-742  
DISK\$VMSMASTER:[FDL.SRC]FDLPARSE.B32;1

Page 13  
(6)

1A	A0	S2	B0	00090	68:	MOVW	R2, 26(XAB)		
	50		A0	000A1		MOVL	4(XAB), XAB	1118	
			A0	11		BRB	5\$		
	6E	0100	8F	3C	000A7	78:	MOVZWL	#256, BYTES	1126
			56	DD	000AC		PUSHL	R6	1130
		04	AE	9F	000AE		PUSHAB	BYTES	
00000000G	00		02	F8	000B1		CALLS	#2, LIB\$FREE_VM	
	09		50	E8	000B8		BLBS	STATUS, 8\$	
00000000G	00		50	DD	000BB		PUSHL	STATUS	1132
	50		01	FB	000BD		CALLS	#1, LIB\$STOP	
			53	D0	000C4	88:	MOVL	STATUS, R0	1135
			04	000C7		RET		1137	

; Routine Size: 200 bytes, Routine Base: \_FDLSCODE + 0074

```
: 622      1138 1 ZSBTTL 'LINE_PARSED'  
: 623      1139 1 GLOBAL ROUTINE FDLS$LINE_PARSED =  
: 624      1140 1 ++  
: 625      1141 1 Functional Description:  
: 626      1142 1  
: 627      1143 1 Main parsing routine. Called by the parse tables it in turn  
: 628      1144 1 calls the appropriate routines to parse the fdl line.  
: 629      1145 1  
: 630      1146 1  
: 631      1147 1 Calling Sequence:  
: 632      1148 1  
: 633      1149 1  
: 634      1150 1  
: 635      1151 1  
: 636      1152 1  
: 637      1153 1  
: 638      1154 1  
: 639      1155 1  
: 640      1156 1  
: 641      1157 1  
: 642      1158 1  
: 643      1159 1  
: 644      1160 1  
: 645      1161 1  
: 646      1162 1  
: 647      1163 1  
: 648      1164 1  
: 649      1165 1  
: 650      1166 1  
: 651      1167 1  
: 652      1168 1  
: 653      1169 1  
: 654      1170 1  
: 655      1171 1  
: 656      1172 1  
: 657      1173 1  
: 658      1174 1  
: 659      1175 1  
: 660      1176 1  
: 661      1177 1  
: 662      1178 1  
: 663      1179 1  
: 664      1180 1  
: 665      1181 1  
: 666      1182 1  
: 667      1183 1  
: 668      1184 1  
: 669      1185 1  
: 670      1186 1  
: 671      1187 2  
: 672      1188 2  
: 673      1189 2  
: 674      1190 2  
: 675      1191 2  
: 676      1192 2  
: 677      1193 2  
: 678      1194 2  
      .fdl$gl_pcall  
      set_area_p  
      set_date_p  
      set_jnl_p  
      set_acl_p  
      set_file_p  
      set_key_p  
      set_record_p  
      set_access_p  
      set_sharing_p  
      set_connect_p  
      not supported V4.0  
      Side Effects:  
      none  
      --  
      BEGIN  
      TPARSE_ARGS;  
      LOCAL  
      STATUS;  
      STATUS = SSS_NORMAL;
```

479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531

1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1207 2  
1208 2  
1209 2  
1210 2  
1211 2  
1212 2  
1213 2  
1214 2  
1215 2  
1216 2  
1217 2  
1218 2  
1219 2  
1220 2  
1221 2  
1222 2  
1223 2  
1224 2  
1225 2  
1226 2  
1227 2  
1228 2  
1229 2  
1230 2  
1231 2  
1232 2  
1233 2  
1234 2  
1235 2  
1236 2  
1237 2  
1238 2  
1239 2  
1240 2  
1241 2  
1242 2  
1243 2  
1244 2  
1245 2  
1246 2  
1247 1

| If we have processed some really bad stuff then dont bother  
| IF .FDLSAB\_CTRL [ FDLSV\_STATUS ] EQLU STS8K\_ERROR  
| THEN  
| RETURN .STATUS;  
| If this is an EDF call then let them process the command  
| IF .FDLSAB\_CTRL [ FDLSV\_PCALL ]  
| THEN  
| STATUS = (.FDL8GL\_PCALL)()  
| ELSE  
| If this is a primary only or line comment call ignore it  
| IF NOT ( .FDLSAB\_CTRL [ FDLSV\_NEWPRI ] OR .FDLSAB\_CTRL [ FDLSV\_LINECMT ] )  
| THEN  
| CASE .FDL8GL\_PRIMARY FROM FDLSC\_ACCESS TO FDLSC\_TITLE OF  
| SET  
| [ FDLSC\_ACCESS ] : SET\_ACCESS\_P();  
| [ FDLSC\_ACL ] : SET\_ACL\_P();  
| [ FDLSC\_AREA ] : SET\_AREA\_P();  
| [ FDLSC\_CONNECT ] : SET\_CONNECT\_P();  
| [ FDLSC\_DATE ] : SET\_DATE\_P();  
| [ FDLSC\_FILE ] : SET\_FILE\_P();  
| [ FDLSC\_JNL ] : SET\_JNL\_P();  
| [ FDLSC\_KEY ] : SET\_KEY\_P();  
| [ FDLSC\_RECORD ] : SET\_RECORD\_P();  
| [ FDLSC\_SHARING ] : SET\_SHARING\_P();  
| [ INRANGE ] : 0; | Catch all for non usefull  
| primaries  
| TES;  
| Clear new primary in case it was set  
| FDLSAB\_CTRL [ FDLSV\_NEWPRI ] = \_CLEAR;  
RETURN .STATUS  
END;

02

63

53	00000000G	00	000C	00000
52		01	9E	00002
03		00	00	00009
		03	ED	0000C
		12		00011
		009E	31	00013
		02	E1	00016
0F	01	A3	00	0001B
		50	00000000G	00
		60	00	FB
		52	00	00022
		50	00	00025
		7E	11	00028
7A	01	63	05	E0
7E	01	A3	01	E0
OE	01	00000000G	00	0002A
0076	0076	0027	001E	0002E
004B	0042	0039	0030	00033
0066	005D	0054	0076	00043
	0076	0076	006F	0004B
				00053

.ENTRY	FDL\$LINE_PARSED, Save R2,R3	1139					
MOVAB	FDL\$AB_CTRL, R3						
MOVL	#1, STATUS	1194					
CMPZV	#0, #3, FDL\$AB_CTRL, #2	1198					
BNEQ	1\$						
BRW	16\$						
BBC	#2, FDL\$AB_CTRL+1, 28	1204					
MOVL	FDL\$GL_PCAEL, R0	1206					
CALLS	#0, (R0)						
MOVL	R0, STATUS						
BRB	13\$						
BBS	#5, FDL\$AB_CTRL, 138	1211					
BBS	#1, FDL\$AB_CTRL+1, 158						
CASEL	FDL\$GL_PRIMARY, #1, #14	1213					
.WORD	48-3\$,-						
	58-3\$,-						
	158-3\$,-						
	158-3\$,-						
	68-3\$,-						
	78-3\$,-						
	88-3\$,-						
	98-3\$,-						
	158-3\$,-						
	108-3\$,-						
	118-3\$,-						
	128-3\$,-						
	148-3\$,-						
	158-3\$,-						
	158-3\$,-						
00000000V 00	00	FB	00059	48:			
	4F	11	00060	CALLS	#0 SET_ACCESS_P	1216	
00000000V 00	00	FB	00062	58:	BRB	15\$	
	46	11	00069	CALLS	#0 SET_ACL_P	1218	
00000000V 00	00	FB	0006B	68:	BRB	15\$	
	3D	11	00072	CALLS	#0 SET_AREA_P	1220	
00000000V 00	00	FB	00074	78:	BRB	15\$	
	34	11	0007B	CALLS	#0 SET_CONNECT_P	1222	
00000000V 00	00	FB	0007D	88:	BRB	15\$	
	2B	11	00084	CALLS	#0 SET_DATE_P	1224	
00000000V 00	00	FB	00086	98:	BRB	15\$	
	22	11	0008D	CALLS	#0 SET_FILE_P	1226	
00000000V 00	00	FB	0008F	108:	BRB	15\$	
	00	FB	00096	CALLS	#0 SET_JNL_P	1228	
00000000V 00	00	FB	00098	118:	BRB	15\$	
	19	11	0009F	CALLS	#0 SET_KEY_P	1230	
00000000V 00	00	FB	000A1	128:	BRB	15\$	
	10	11	000A8	CALLS	#0 SET_RECORD_P	1232	
00000000V 00	00	FB	000AA	138:	BRB	15\$	
	07	11	000B1	CALLS	#0 SET_SHARING_P	1234	
63	20	8A	000B1	158:	BICB2	#32, FDL\$AB_CTRL	1243
50	52	D0	000B4	168:	MOVL	STATUS, R0	1245
		04	000B7	RET			1247

: Routine Size: 184 bytes. Routine Base: \_FDL\$CODE + 013C

```
533 1248 1 ISBTTL 'SET_AREA_P'  
534 1249 1 ROUTINE SET_AREA_P : NOVALUE =  
535 1250 1 ++  
536 1251 1 Functional Description:  
537 1252 1 Fill in the blanks for the allocation xab  
538 1253 1  
539 1254 1 Calling Sequence:  
540 1255 1 set_area_p()  
541 1256 1  
542 1257 1 Input Parameters:  
543 1258 1 none  
544 1259 1  
545 1260 1 Implicit Inputs:  
546 1261 1  
547 1262 1 fdLSsecondary - Secondary code  
548 1263 1  
549 1264 1 Output Parameters:  
550 1265 1 none  
551 1266 1  
552 1267 1 Implicit Outputs:  
553 1268 1 none  
554 1269 1  
555 1270 1 Routine Value:  
556 1271 1 none  
557 1272 1  
558 1273 1 Routines Called:  
559 1274 1  
560 1275 1 allocate_xab  
561 1276 1  
562 1277 1 Side Effects:  
563 1278 1 none  
564 1279 1  
565 1280 1  
566 1281 1  
567 1282 1  
568 1283 1 --  
569 1284 1  
570 1285 2 BEGIN  
571 1286 2  
572 1287 2 ! To avoid some duplication of code ...  
573 1288 2 ! Find out if there is a current xab if not then get one  
574 1289 2 ! OR If the current xab is not the same type or number of what we want  
575 1290 2 then get a new one  
576 1291 2  
577 1292 3 IF ( .CURRENT_XAB EQLU 0  
578 1293 3 THEN 1  
579 1294 3 ELSE  
580 1295 3 IF ( .CURRENT_XAB [ XABSB_COD ] NEQ XABSC_ALL ) OR  
581 1296 4 ( .CURRENT_XAB [ XABSB_AID ] NEQ .FDL$GL_PRINUM )  
582 1297 3 THEN 1  
583 1298 3 ELSE 0 )  
584 1299 2 THEN  
585 1300 2 BEGIN  
586 1301 2  
587 1302 2 ! Allocate memory for the new xab  
588 1303 2  
589 1304 3 ALLOCATE_XAB ( XABSC_ALL, .FDL$GL_PRINUM );
```

```
590      1305 3
591      1306 3
592      1307 3
593      1308 3
594      1309 3
595      1310 3
596      1311 3
597      1312 3
598      1313 3
599      1314 3
600      1315 3
601      1316 4
602      1317 4
603      1318 4
604      1319 4
605      1320 4
606      1321 4
607      1322 4
608      1323 4
609      1324 4
610      1325 4
611      1326 4
612      1327 4
613      1328 4
614      1329 4
615      1330 4
616      1331 4
617      1332 4
618      1333 4
619      1334 4
620      1335 4
621      1336 4
622      1337 4
623      1338 4
624      1339 4
625      1340 4
626      1341 4
627      1342 4
628      1343 4
629      1344 4
630      1345 4
631      1346 3
632      1347 3
633      1348 3
634      1349 3
635      1350 3
636      1351 3
637      1352 2
638      1353 2
639      1354 2
640      1355 2
641      1356 2
642      1357 2
643      1358 2
644      1359 2
645      1360 2
646      1361 2

      ; Set the area number in the xab
      CURRENT_XAB [ XABSB_AID ] = .FDL$GL_PRINUM;

      ; If this is area 0 then copy the allocation etc. from the fab (this
      ; is because using areas override the fab allocation and this
      ; makes it look like it doesn't)

      IF .CURRENT_XAB [ XABSB_AID ] EQLU 0
      THEN
        BEGIN

          ; Copy Allocation, Bucket size and Extention
          CURRENT_XAB [ XABSL_ALQ ] = .FDL$AB_PARSED_FAB [ FABSL_ALQ ];
          CURRENT_XAB [ XABSB_BKZ ] = .FDL$AB_PARSED_FAB [ FABSB_BKS ];
          CURRENT_XAB [ XABSW_DEQ ] = .FDL$AB_PARSED_FAB [ FABSW_DEQ ];
          CURRENT_XAB [ XABSL_ALQ ] = .FDL$AB_PARSED_FAB [ FABSL_ALQ ];

          IF .FDL$AB_PARSED_FAB [ FABSB_BKS ] NEQU 0
          THEN
            FDLSAB_AREA_BKZ [ 0 ] = .FDL$AB_PARSED_FAB [ FABSB_BKS ]
          ELSE
            FDLSAB_AREA_BKZ [ 0 ] = BUCKET_DEFAULT;

          ; Also get the duplicated contiguous options:
          ; Contiguous best try
          IF .FDL$AB_PARSED_FAB [ FAB$V_CBT ]
          THEN
            CURRENT_XAB [ XAB$V_CBT ] = _SET;
          ; Contiguous
          IF .FDL$AB_PARSED_FAB [ FAB$V_CTG ]
          THEN
            CURRENT_XAB [ XAB$V_CTG ] = _SET
          END
        ELSE
          ; Count this area
          HIGHEST_AREA_NO = .HIGHEST_AREA_NO + 1
        END;

      ; Set the fields in the area xab
      CASE .FDL$GL_SECONDARY FROM FDLSC_ALLOC TO FDLSC_VOLU OF
      SET
        [ FDLSC_ALLOC ] : CURRENT_XAB [ XABSL_ALQ ] = .FDL$GL_NUMBER;
        [ FDLSC_BTCONT ]: CURRENT_XAB [ XAB$V_CBT ] = .FDL$GL_SWITCH;
```

647 1362 3 [ FDLSC\_BKT ] : BEGIN  
648 1363 CURRENT\_XAB [ XABSB\_BKZ ] = .FDL\$GL\_NUMBER;  
649 1364 ; Fill in the table for figuring fill numbers latter  
650 1365 FDLSAB\_AREA\_BKZ [ .FDL\$GL\_PRINUM ] = .FDL\$GL\_NUMBER  
651 1366  
652 1367  
653 1368  
654 1369  
655 1370  
656 1371  
657 1372 [ FDLSC\_CONTG ] : CURRENT\_XAB [ XABSV\_CTG ] = .FDL\$GL\_SWITCH;  
658 1373 [ FDLSC\_EXACT ] : CURRENT\_XAB [ XABSV\_HRD ] = .FDL\$GL\_SWITCH;  
659 1374 [ FDLSC\_EXTND ] : CURRENT\_XAB [ XABSW\_DEQ ] = .FDL\$GL\_NUMBER;  
660 1375 [ FDLSC\_POSI ] : CASE .FDL\$GL\_QUALIFIER FROM  
661 1376 FDLSC\_ANYPOS TO FDLSC\_VIRPOS OF  
662 1377 SET [ FDLSC\_ANYPOS ] : CURRENT\_XAB [ XABSV\_ONC ] = \_SET;  
663 1378 [ FDLSC\_CLUSPOS ] : CURRENT\_XAB [ XABSV\_ONC ] = \_SET;  
664 1379 [ FDLSC\_CYLPOS ] : BEGIN  
665 1380 CURRENT\_XAB [ XABSB\_ALN ] = XABSC\_CYL;  
666 1381 CURRENT\_XAB [ XABSL\_LOC ] = .FDL\$GL\_NUMBER  
667 1382 END;  
668 1383 [ FDLSC\_FIDPOS ] : BEGIN  
669 1384 CURRENT\_XAB [ XABSW\_RF10 ] = .FDL\$GL\_FID1;  
670 1385 CURRENT\_XAB [ XABSW\_RF12 ] = .FDL\$GL\_FID2;  
671 1386 CURRENT\_XAB [ XABSW\_RF14 ] = .FDL\$GL\_FID3  
672 1387 END;  
673 1388 [ FDLSC\_FNMPPOS ] : BEGIN  
674 1389 FIND ID();  
675 1390 CURRENT\_XAB [ XABSW\_RF10 ] = .FDL\$GL\_FID1;  
676 1391 CURRENT\_XAB [ XABSW\_RF12 ] = .FDL\$GL\_FID2;  
677 1392 CURRENT\_XAB [ XABSW\_RF14 ] = .FDL\$GL\_FID3  
678 1393 END;  
679 1394 [ FDLSC\_LOGPOS ] : BEGIN  
680 1395 CURRENT\_XAB [ XABSB\_ALN ] = XABSC\_LBN;  
681 1396 CURRENT\_XAB [ XABSL\_LOC ] = .FDL\$GL\_NUMBER  
682 1397 END;  
683 1398 [ FDLSC\_NOPOS ] : CURRENT\_XAB [ XABSB\_ALN ] = \_CLEAR;  
684 1399 [ FDLSC\_VIRPOS ] : BEGIN  
685 1400 CURRENT\_XAB [ XABSB\_ALN ] = XABSC\_VBN;  
686 1401 CURRENT\_XAB [ XABSL\_LOC ] = .FDL\$GL\_NUMBER  
687 1402 END;  
688 1403 TES:  
689 1404 [ FDLSC\_VOLU ] : BEGIN  
690 1405 CURRENT\_XAB [ XABSW\_VOL ] = .FDL\$GL\_NUMBER;  
691 1406  
692 1407  
693 1408  
694 1409  
695 1410  
696 1411  
697 1412  
698 1413  
699 1414  
700 1415  
701 1416  
702 1417  
703 1418

```

704    1419 3
705    1420
706    1421
707    1422
708    1423
709    1424
710    1425
711    1426
712    1427
713    1428
714    1429
715    1430
716    1431 1
      TES:
      RETURN
      END;

```

```

| If the guy didn't give any placement do it for him
| IF .CURRENT_XAB [ XABSB_ALN ] EQLU _CLEAR
| THEN
|   CURRENT_XAB [ XABSB_ALN ] = XABSC_LBN;
END;

```

00FC 00000 SET_AREA_P:						
				.WORD	Save R2,R3,R4,R5,R6,R7	1249
	57	00000000G	00	9E 00002	FDLSAB_AREA_BKZ, R7	
	56	00000000G	00	9E 00009	FDLSGL_SWITCH, R6	1292
	55	00000000G	00	9E 00010	FDLSGL_PRINUM, R5	
	54	00000000G	00	9E 00017	FDLSGL_NUMBER, R4	
	53	00000000	00	9E 0001E	CURRENT_XAB, R3	
	50		63	D0 00025	MOVL	
			0D	13 00028	BEQL	1S
	14		60	91 0002A	CMPB	(R0), #20
			08	12 0002D	BNEQ	1S
			00	E0 0002F	CMPZV	#0, #8, 23(R0), FDLSGL_PRINUM
			58	13 00035	BEQL	6S
			65	DD 00037	PUSHL	FDLSGL_PRINUM
			14	DD 00039	PUSHL	#20
	65	00000000V	00	02 FB 0003B	CALLS	#2, ALLOCATE_XAB
			51	63 D0 00042	MOVL	CURRENT_XAB, R1
	17	A1	65	90 00045	MOVB	FDLSGL_PRINUM, 23(R1)
			41	12 00049	BNEQ	5S
		50 00000000G	00	D0 0004B	MOVL	FDLSAB_PARSED_FAB, R0
	10	A1	10	A0 D0 00052	MOVL	16(R0), 16(R1)
	16	A1	3E	A0 90 00057	MOVB	62(R0), 22(R1)
	14	A1	14	A0 B0 0005C	MOVW	20(R0), 20(R1)
	10	A1	10	A0 D0 00061	MOVL	16(R0), 16(R1)
		52	67	D0 00066	MOVL	FDLSAB_AREA_BKZ, R2
			3E	A0 95 00069	TSTB	62(R0)
			06	13 0006C	BEQL	2S
			62	A0 90 0006E	MOVB	62(R0), (R2)
			03	11 00072	BRB	3S
	04	06	62 A0	02 90 00074	MOVB	#2, (R2)
		08	A1	05 E1 00077	BBC	#5, 6(R0), 4S
	0A	06	A0	20 88 0007C	BISB2	#32, 8(R1), 4S
		08	A1	04 E1 00080	BBC	#4, 6(R0), 6S
			80	8F 88 00085	BISB2	#128, 8(R1)
			03	11 0008A	BRB	6S
			FC A3	96 0008C	INCBL	HIGHEST AREA NO
0034	07	18 00000000G	00	CF 0008F	CASEL	FDLSGL_SECONDARY, #27, #7
00B8	0022	0018	0010	00097	.WORD	8S-7S,-
	0050	0048	003E	0009F		9S-7S,-



FDLPARSE  
V04-000

VAX-11 FDL Utilities  
SET\_AREA\_P

K 6  
16-Sep-1984 01:50:08  
16-Sep-1984 12:31:19

VAX-11 BLiss-32 V4.0-742  
DISK\$VMSMASTER:[FDL.SRC]FDLPARSE.B32;1

Page 22  
(8)

0A	50	09	63	04 0014E	248:	RET	CURRENT_XAB, R0
	A0		64	D0 0014F		MOVL	FDL\$GL_NUMB <small>E</small> R, 10(R0)
			A0	B0 00152		MOVW	9(R0)
			95	00156		TSTB	25S
			04	12 00159		BNEQ	#2, 9(R0)
09	A0		02	90 0015B		MOVB	
			04	0015F	258:	RET	

; 1378  
; 1417  
; 1421  
; 1423  
; 1431

: Routine Size: 352 bytes, Routine Base: \_FDL\$CODE + 01F4

```
; 718      1 XSBTTL 'SET_DATE_P'  
; 719      1 ROUTINE SET_DATE_P : NOVALUE =  
; 720      1 !++  
; 721      1 ! Functional Description:  
; 722      1     Fill in the blanks for the revision date and time xab  
; 723      1  
; 724      1     Calling Sequence:  
; 725      1         set_date_p()  
; 726      1  
; 727      1     Input Parameters:  
; 728      1         none  
; 729      1  
; 730      1     Implicit Inputs:  
; 731      1  
; 732      1         fdl$secondary - Secondary code  
; 733      1  
; 734      1     Output Parameters:  
; 735      1         none  
; 736      1  
; 737      1     Implicit Outputs:  
; 738      1         none  
; 739      1  
; 740      1     Routine Value:  
; 741      1         none  
; 742      1  
; 743      1     Routines Called:  
; 744      1         sys$bintim  
; 745      1  
; 746      1     Side Effects:  
; 747      1         none  
; 748      1  
; 749      1  
; 750      1  
; 751      1  
; 752      1  
; 753      1 !--  
; 754      1  
; 755      2     BEGIN  
; 756      2  
; 757      2     ! See which xab we need  
; 758      2  
; 759      2     IF .FDL$GL_SECONDARY EQLU FDLSC_REV  
; 760      2     THEN  
; 761      2         BEGIN  
; 762      2  
; 763      2         ! If the revision xab has not been connected then connect it  
; 764      2  
; 765      2         IF .REVISION_XAB EQLU 0  
; 766      2         THEN  
; 767      2  
; 768      2         ! Allocate the xab an enter it into the chain  
; 769      2  
; 770      2         REVISION_XAB = ALLOCATE_XAB ( XABSC_RDT, 0 )  
; 771      2  
; 772      2  
; 773      2         END  
; 774      2     ELSE
```

```

775 1489 2      | If the date xab has not been allocated then get one
776 1490 2
777 1491 2
778 1492 2
779 1493 2
780 1494 2
781 1495 2
782 1496 2
783 1497 2
784 1498 2
785 1499 2
786 1500 2
787 1501 2
788 1502 2
789 1503 2
790 1504 2
791 1505 2
792 1506 2
793 1507 2
794 1508 2
795 1509 2
796 1510 2
797 1511 2
798 1512 2
799 1513 2
800 1514 2
801 1515 2
802 1516 2
803 1517 2
804 1518 2
805 1519 2
806 1520 2
807 1521 2
808 1522 2
809 1523 2
810 1524 2
811 1525 2
812 1526 1

    ! If the date xab has not been allocated then get one
    IF .DATE_XAB EQLU 0
    THEN
        ! Allocate the xab an enter it into the chain
        DATE_XAB = ALLOCATE_XAB ( XABSC_DAT, 0 );
    ! Fill in the correct field
    CASE .FDL$GL_SECONDARY FROM FDL$C_BACKUP TO FDL$C_REV OF
    SET
        [ FDL$C_BACKUP ]: BEGIN
            DATE_XAB [ XABSL_BDT0 ] = .FDL$AL_DATE_TIME [ 0 ];
            DATE_XAB [ XABSL_BDT4 ] = .FDL$AL_DATE_TIME [ 1 ];
        END;
        [ FDL$C_CREAT ] : BEGIN
            DATE_XAB [ XABSL_CDT0 ] = .FDL$AL_DATE_TIME [ 0 ];
            DATE_XAB [ XABSL_CDT4 ] = .FDL$AL_DATE_TIME [ 1 ];
        END;
        [ FDL$C_EXPR ] : BEGIN
            DATE_XAB [ XABSL_EDT0 ] = .FDL$AL_DATE_TIME [ 0 ];
            DATE_XAB [ XABSL_EDT4 ] = .FDL$AL_DATE_TIME [ 1 ];
        END;
        [ FDL$C_REV ] : BEGIN
            REVISION_XAB [ XABSL_RDT0 ] = .FDL$AL_DATE_TIME [ 0 ];
            REVISION_XAB [ XABSL_RDT4 ] = .FDL$AL_DATE_TIME [ 1 ];
        END;
    TES;
    RETURN
    END;

```

; Routine Size: 141 bytes,      Routine Base: \_FDLSCODE + 0354

```
: 814      1527 1 ISBTTL 'SET_JNL_P'  
: 815      1528 1 ROUTINE SET_JNL_P : NOVALUE =  
: 816      1529 1 ++  
: 817      1530 1 Functional Description:  
: 818      1531 1     Fill in the blanks for the journal xab  
: 819      1532 1 Calling Sequence:  
: 820      1533 1     set_jnl_p()  
: 821      1534 1 Input Parameters:  
: 822      1535 1     none  
: 823      1536 1 Implicit Inputs:  
: 824      1537 1     fdlssecondary - Secondary code  
: 825      1538 1 Output Parameters:  
: 826      1539 1     none  
: 827      1540 1 Implicit Outputs:  
: 828      1541 1     none  
: 829      1542 1 Routine Value:  
: 830      1543 1     none  
: 831      1544 1 Routines Called:  
: 832      1545 1     none  
: 833      1546 1 Side Effects:  
: 834      1547 1     none  
: 835      1548 1  
: 836      1549 1  
: 837      1550 1  
: 838      1551 1  
: 839      1552 1  
: 840      1553 1  
: 841      1554 1  
: 842      1555 1  
: 843      1556 1  
: 844      1557 1  
: 845      1558 1  
: 846      1559 1  
: 847      1560 1  
: 848      1561 1  
: 849      1562 1 --  
: 850      1563 1  
: 851      1564 2 BEGIN  
: 852      1565 2  
: 853      1566 2     If the xab has not been connected, then connect it  
: 854      1567 2  
: 855      1568 2 IF .JNL_XAB EQLU 0  
: 856      1569 2 THEN  
: 857      1570 2     Allocate the xab and enter it into the chain  
: 858      1571 2     JNL_XAB = ALLOCATE_XAB ( XABSC_JNL, 0 );  
: 859      1572 2  
: 860      1573 2  
: 861      1574 2     Fill in the correct field  
: 862      1575 2  
: 863      1576 2 CASE .FDLSQL_SECONDARY FROM FDLSC_AFTIM TO FDLSC_RU OF  
: 864      1577 2 SET  
: 865      1578 2     [ FDLSC_AFTIM ] : JNL_XAB [ XABSV_AI ] = .FDLSQL_SWITCH;  
: 866      1579 2  
: 867      1580 2     [ FDLSC_AFTNAM ] : BEGIN  
: 868      1581 2  
: 869      1582 2     Allocate a buffer for the string and copy to it  
: 870      1583 2
```

871 1584 :  
872 1585 :  
873 1586 :  
874 1587 : JNL\_XAB [ XABSL\_AIA ] =  
875 1588 : FDLS\$GET\_VMT.FDLSAB\_STRING [ DSCSW\_LENGTH ] );  
876 1589 :  
877 1590 :  
878 1591 : CHSMOVE( .FDLSAB\_STRING [ DSCSW\_LENGTH ],  
879 1592 : .FDLSAB\_STRING [ DSCSA\_POINTER ],  
880 1593 : .JNL\_XAB [ XABSL\_AIA ] );  
881 1594 :  
882 1595 : JNL\_XAB [ XABSB\_AIS ] =  
883 1596 : .FD[SAB\_STRING [ DSCSW\_LENGTH ]  
884 1597 : END;  
885 1598 : [ FDLSC\_AUDIT ] : JNL\_XAB [ XABSV\_AT ] = .FDLSGL\_SWITCH;  
886 1599 :  
887 1600 : [ FDLSC\_AUDNAM ] : BEGIN  
888 1601 : | Allocate a buffer for the string and copy to it  
889 1602 : JNL\_XAB [ XABSL\_ATA ] =  
890 1603 : FDLS\$GET\_VMT.FDLSAB\_STRING [ DSCSW\_LENGTH ] );  
891 1604 :  
892 1605 : CHSMOVE( .FDLSAB\_STRING [ DSCSW\_LENGTH ],  
893 1606 : .FDLSAB\_STRING [ DSCSA\_POINTER ],  
894 1607 : .JNL\_XAB [ XABSL\_ATA ] );  
895 1608 :  
896 1609 : JNL\_XAB [ XABSB\_ATS ] =  
897 1610 : .FD[SAB\_STRING [ DSCSW\_LENGTH ]  
898 1611 : END;  
899 1612 :  
900 1613 : [ FDLSC\_BEFIM ] : JNL\_XAB [ XABSV\_BI ] = .FDLSGL\_SWITCH;  
901 1614 :  
902 1615 : [ FDLSC\_BEFNAM ] : BEGIN  
903 1616 : | Allocate a buffer for the string and copy to it  
904 1617 : JNL\_XAB [ XABSL\_BIA ] =  
905 1618 : FDLS\$GET\_VMT.FDLSAB\_STRING [ DSCSW\_LENGTH ] );  
906 1619 :  
907 1620 : CHSMOVE( .FDLSAB\_STRING [ DSCSW\_LENGTH ],  
908 1621 : .FDLSAB\_STRING [ DSCSA\_POINTER ],  
909 1622 : .JNL\_XAB [ XABSL\_BIA ] );  
910 1623 :  
911 1624 : JNL\_XAB [ XABSB\_BIS ] =  
912 1625 : .FD[SAB\_STRING [ DSCSW\_LENGTH ]  
913 1626 : END;  
914 1627 :  
915 1628 : [ FDLSC\_RU ] : BEGIN  
916 1629 : | Set the recovery unit bit according to what  
917 1630 : | was specified  
918 1631 : JNL\_XAB [ XABSV\_RU ] = \_CLEAR;  
919 1632 : JNL\_XAB [ XABSV\_ONLY\_RU ] = \_CLEAR;  
920 1633 : JNL\_XAB [ XABSV\_NEVER\_RU ] = \_CLEAR;  
921 1634 :  
922 1635 : IF .FDLSGL\_QUALIFIER EQLU FDLSC\_IF\_IN  
923 1636 : THEN  
924 1637 : JNL\_XAB [ XABSV\_RU ] = \_SET  
925 1638 : ELSE IF .FDLSGL\_QUALIFIER EQLU FDLSC\_NEQ  
926 1639 : THEN  
927 1640 :  
FDI  
VO

928	1641	7
929	1642	7
930	1643	7
931	1644	7
932	1645	7
933	1646	7
934	1647	7
935	1648	7
936	1649	7
937	1650	7
938	1651	7
939	1652	7
940	1653	7

```
JNL_XAB [ XABSV_ONLY_RU ] = _SET  
ELSE IF .FDLSQL_QUALIFIER EQLU FDLSC_NEVER  
THEN  
    JNL_XAB [ XABSV_NEVER_RU ] = _SET;  
END:
```

TES:  
RETUR  
END:

0FFC 00000 SET\_JNL\_P:

	20 B6		60	57 28 0008C	MOV C3	R7, (R0), #32(R6)	: 1608	
		1C	A6	57 90 00091	MOVB	R7, 28(R6)	: 1607	
08 A2	01	02		6A F0 00096	78:	INSV	FDL\$GL_SWITCH, #2, #1, 8(R2)	: 1611
			7E	04 00095		RET		: 1617
			6B	68 3C 0009D	88:	MOVZWL	FDL\$AB_STRING, -(SP)	: 1619
		10	A2	01 FB 000A0		CALLS	#1, FD[\$\$GET_VM	: 1620
			57	50 D0 000A3		MOVL	RO, 16(R2)	: 1621
			50	68 3C 000A7		MOVZWL	FDL\$AB_STRING, R7	: 1624
	10 B6		04	A8 D0 000AA		MOVL	FDL\$AB_STRING+4, RO	: 1623
			56	69 D0 000AE		MOVL	JNL_XAB, R6	: 1631
			60	57 28 000B1		MOV C3	R7, -(RO), #16(R6)	: 1633
			60	57 90 000B6		MOVB	R7, 12(R6)	: 1635
			51	08 A2 9E 000B8	95:	RET		: 1637
			61	23 8A 000BF		MOVAB	8(R2), R1	: 1639
			50 00000000G	00 00 000C2		BICB2	#35, (R1)	: 1641
			13	50 D1 000C9		MOVL	FDL\$GL_QUALIFIER, RO	: 1643
			61	04 12 000CC		CMPL	RO, #19	: 1645
			02	88 000CE		BNEQ	10\$	: 1653
			14	04 000D1		BISB2	#2, (R1)	
			61	50 D1 000D2	108:	RET		
			04	12 000D5		CMPL	RO, #20	
			01	88 000D7		BNEQ	11\$	
			15	04 000DA	118:	BISB2	#1, (R1)	
			03	50 D1 000DB		RET		
			20	12 000DE		CMPL	RO, #21	
			04	88 000E0		BNEQ	12\$	
			04 000E3	128:		BISB2	#32, (R1)	
						RET		

: Routine Size: 228 bytes. Routine Base: \_FDLSCODE + 03E1

```
: 942  
: 943 1654 1 %SBTTL 'SET_ACL_P'  
: 944 1655 1 ROUTINE SET_ACL_P : NOVALUE =  
: 945 1656 1 **  
: 946 1657 1 Functional Description:  
: 947 1658 1      Fill in the blanks for the ACL xab  
: 948 1659 1 Calling Sequence:  
: 949 1660 1      set_acl_p()  
: 950 1661 1 Input Parameters:  
: 951 1662 1      none  
: 952 1663 1 Implicit Inputs:  
: 953 1664 1      fdL$secondary - Secondary code  
: 954 1665 1 Output Parameters:  
: 955 1666 1      none  
: 956 1667 1 Implicit Outputs:  
: 957 1668 1      none  
: 958 1669 1 Routine Values:  
: 959 1670 1      none  
: 960 1671 1 Routines Called:  
: 961 1672 1      none  
: 962 1673 1 Side Effects:  
: 963 1674 1      none  
: 964 1675 1      none  
: 965 1676 1  
: 966 1677 1  
: 967 1678 1  
: 968 1679 1  
: 969 1680 1  
: 970 1681 1  
: 971 1682 1  
: 972 1683 1  
: 973 1684 1  
: 974 1685 1  
: 975 1686 1  
: 976 1687 1  
: 977 1688 1  
: 978 1689 1 --  
: 979 1690 1 BEGIN  
: 980 1691 2 !  
: 981 1692 2 : nop until there exists an ACLXAB  
: 982 1693 2  
: 983 1694 2 RETURN  
: 984 1695 2  
: 985 1696 2  
: 987 1697 1 END:
```

0000 00000 SET\_ACL\_P:  
04 00002 WORD Save nothing  
RET

: 1655  
: 1697

: Routine Size: 3 bytes, Routine Base: \_FDLSCODE + 04C5

```
987 1698 1 %SBTTL "SET FILE P"
988 1699 1 ROUTINE SET_FILE_P : NOVALUE =
989 1700 1 ++
990 1701 1 Functional Description:
991 1702 1 Fill in the blanks for the fab
992 1703 1
993 1704 1 Calling Sequence:
994 1705 1     set_file_p()
995 1706 1
996 1707 1 Input Parameters:
997 1708 1     none
998 1709 1
999 1710 1 Implicit Inputs:
1000 1711 1     fdI$secondary - Secondary code
1001 1712 1
1002 1713 1 Output Parameters:
1003 1714 1     none
1004 1715 1 Implicit Outputs:
1005 1716 1     none
1006 1717 1 Routine Value:
1007 1718 1     SSS_NORMAL or error from set_prot
1008 1719 1
1009 1720 1 Routines Called:
1010 1721 1     fdI$Sget_vm
1011 1722 1     set_prot
1012 1723 1
1013 1724 1 Side Effects:
1014 1725 1     none
1015 1726 1
1016 1727 1
1017 1728 1
1018 1729 1
1019 1730 1
1020 1731 1
1021 1732 1
1022 1733 1
1023 1734 1
1024 1735 1
1025 1736 1
1026 1737 2
1027 1738 2
1028 1739 2
1029 1740 2
1030 1741 2
1031 1742 2
1032 1743 2
1033 1744 2
1034 1745 2
1035 1746 2
1036 1747 2
1037 1748 2
1038 1749 2
1039 1750 2
1040 1751 2
1041 1752 2
1042 1753 2
1043 1754 2

-- BEGIN
REGISTER
    PARSED_FAB : REF BLOCK [ .BYTE ];
PARSED_FAB = .FDL$AB_PARSED_FAB;
! Set the fab according to the secondary parsed
SELECT .FDL$GL_SECONDARY OF
SET
    [ FDLSC_ALL ] : PARSED_FAB [ FABSL_ALQ ] = .FDL$GL_NUMBER;
    [ FDLSC_BKTUP ] : 0;
    [ FDLSC_BTG ] : PARSED_FAB [ FABSV_CBT ] = .FDL$GL_SWITCH;
    [ FDLSC_BKTSIZ ]: BEGIN
```

: 1044 1755 3  
: 1045 1756 2  
: 1046 1757 2  
: 1047 1758 2  
: 1048 1759 2  
: 1049 1760 2  
: 1050 1761 2  
: 1051 1762 2  
: 1052 1763 2  
: 1053 1764 2  
: 1054 1765 2  
: 1055 1766 2  
: 1056 1767 2  
: 1057 1768 2  
: 1058 1769 2  
: 1059 1770 2  
: 1060 1771 2  
: 1061 1772 2  
: 1062 1773 2  
: 1063 1774 2  
: 1064 1775 2  
: 1065 1776 2  
: 1066 1777 2  
: 1067 1778 2  
: 1068 1779 2  
: 1069 1780 2  
: 1070 1781 2  
: 1071 1782 2  
: 1072 1783 2  
: 1073 1784 2  
: 1074 1785 2  
: 1075 1786 2  
: 1076 1787 2  
: 1077 1788 2  
: 1078 1789 2  
: 1079 1790 2  
: 1080 1791 2  
: 1081 1792 2  
: 1082 1793 2  
: 1083 1794 2  
: 1084 1795 2  
: 1085 1796 2  
: 1086 1797 2  
: 1087 1798 2  
: 1088 1799 2  
: 1089 1800 2  
: 1090 1801 2  
: 1091 1802 2  
: 1092 1803 2  
: 1093 1804 2  
: 1094 1805 2  
: 1095 1806 2  
: 1096 1807 2  
: 1097 1808 2  
: 1098 1809 2  
: 1099 1810 2  
: 1100 1811 2

PARSED\_FAB [ FABSB\_BKS ] = .FDL\$GL\_NUMBER;  
| Stuff the bucket size into the array for latter  
FDLSAB\_AREA\_BKZ [ 0 ] = .FDL\$GL\_NUMBER  
END;  
[ FDLSC\_CLUSIZ ]: 0;  
[ FDLSC\_FCTX ] : PARSED\_FAB [ FABSL\_CTX ] = .FDL\$GL\_NUMBER;  
[ FDLSC\_CONT ] : PARSED\_FAB [ FABSV\_CTG ] = .FDL\$GL\_SWITCH;  
[ FDLSC\_CIF ] : PARSED\_FAB [ FABSV\_CIF ] = .FDL\$GL\_SWITCH;  
[ FDLSC\_DFNAM ] : BEGIN  
| Allocate a buffer for the string and copy it into it  
PARSED\_FAB [ FABSL\_DNA ] =  
FDL\$SGT\_VM( .FDL\$AB\_STRING [ DSC\$W\_LENGTH ] );  
CHSMOVE( .FDL\$AB\_STRING [ DSC\$W\_LENGTH ] ,  
.FDL\$AB\_STRING [ DSC\$A\_POINTER ] ,  
.PARSED\_FAB [ FABSL\_DNA ] );  
PARSED\_FAB [ FABSB\_DNS ] =  
.FDL\$AB\_STRING [ DSC\$W\_LENGTH ]  
END;  
[ FDLSC\_DEFWRIT ] : PARSED\_FAB [ FABSV\_DFW ] = .FDL\$GL\_SWITCH;  
[ FDLSC\_DOC ] : PARSED\_FAB [ FABSV\_DLT ] = .FDL\$GL\_SWITCH;  
[ FDLSC\_DIR ] : PARSED\_FAB [ FABSV\_TMP ] = .FDL\$GL\_SWITCH;  
: not supported V4.0  
[ FDLSC\_EODEL ] : PARSED\_FAB [ FABSV\_EDL ] = .FDL\$GL\_SWITCH;  
[ FDLSC\_EXTN ] : PARSED\_FAB [ FABSW\_DEQ ] = .FDL\$GL\_NUMBER;  
[ FDLSC\_GBC ] : PARSED\_FAB [ FABSW\_GBC ] = .FDL\$GL\_NUMBER;  
[ FDLSC\_MTBLISIZ]: PARSED\_FAB [ FABSW\_BLS ] = .FDL\$GL\_NUMBER;  
[ FDLSC\_MTCP ] : PARSED\_FAB [ FABSV\_POS ] = .FDL\$GL\_SWITCH;  
[ FDLSC\_MTNEF ] : PARSED\_FAB [ FABSV\_NEF ] = .FDL\$GL\_SWITCH;  
[ FDLSC\_MTPRO ] : SET\_PROT();  
[ FDLSC\_MTREW ] : PARSED\_FAB [ FABSV\_RWD ] = .FDL\$GL\_SWITCH;  
[ FDLSC\_MTRWC ] : PARSED\_FAB [ FABSV\_RWC ] = .FDL\$GL\_SWITCH;

: 1101 1812 2 [ FDLSC\_MAXREC ] : PARSED\_FAB [ FABSL\_MRN ] = .FDL\$GL\_NUMBER;  
: 1102 1813 2  
: 1103 1814 2 [ FDLSC\_MAXVER ] : PARSED\_FAB [ FABSV\_MXV ] = .FDL\$GL\_SWITCH;  
: 1104 1815 2  
: 1105 1816 2 [ FDLSC\_NAME ] : BEGIN  
: 1106 1817 2     ! Check for non-null name string  
: 1107 1818 2  
: 1108 1819 2     IF .FDLSAB\_STRING [ DSCSW\_LENGTH ] NEQ 0  
: 1109 1820 2       THEN  
: 1110 1821 4        BEGIN  
: 1111 1822 4           ! Allocate a buffer for the string and copy it  
: 1112 1823 4  
: 1113 1824 4           PARSED\_FAB [ FABSL\_FNA ] =  
: 1114 1825 4            FDL\$SGET\_VM( .FDLSAB\_STRING [ DSCSW\_LENGTH ] );  
: 1115 1826 4  
: 1116 1827 4  
: 1117 1828 4           CHSMOVE( .FDLSAB\_STRING [ DSCSW\_LENGTH ],  
: 1118 1829 4            .FDLSAB\_STRING [ DCSA\_POINTER ],  
: 1119 1830 3            .PARSED\_FAB [ FABSL\_FNA ] );  
: 1120 1831 3  
: 1121 1832 3  
: 1122 1833 2  
: 1123 1834 2  
: 1124 1835 2 [ FDLSC\_NFS ] : PARSED\_FAB [ FABSV\_NFS ] = .FDL\$GL\_SWITCH;  
: 1125 1836 2  
: 1126 1837 2 [ FDLSC\_ORG ] : PARSED\_FAB [ FABSB\_ORG ] = .FDL\$GL\_QUALIFIER;  
: 1127 1838 2  
: 1128 1839 2 [ FDLSC\_OFP ] : PARSED\_FAB [ FABSV\_OFP ] = .FDL\$GL\_SWITCH;  
: 1129 1840 2  
: 1130 1841 2 [ FDLSC\_OWNER ] : SET\_PROT();  
: 1131 1842 2  
: 1132 1843 2 [ FDLSC\_POC ] : PARSED\_FAB [ FABSV\_SPL ] = .FDL\$GL\_SWITCH;  
: 1133 1844 2  
: 1134 1845 2 [ FDLSC\_PROT ] : SET\_PROT();  
: 1135 1846 2  
: 1136 1847 2 [ FDLSC\_READC ] : PARSED\_FAB [ FABSV\_RCK ] = .FDL\$GL\_SWITCH;  
: 1137 1848 2  
: 1138 1849 2 [ FDLSC\_REVISN ] : BEGIN  
: 1139 1850 2     ! If the revision xab has not been connected then connect it  
: 1140 1851 2  
: 1141 1852 2  
: 1142 1853 2     IF .REVISION\_XAB EQLU 0  
: 1143 1854 2       THEN  
: 1144 1855 2  
: 1145 1856 2  
: 1146 1857 2  
: 1147 1858 2     ! Allocate the xab an enter it into the chain  
: 1148 1859 2  
: 1149 1860 2     REVISION\_XAB = ALLOCATE\_XAB ( XABSC\_RDT, 0 );  
: 1150 1861 2  
: 1151 1862 2  
: 1152 1863 2  
: 1153 1864 2 [ FDLSC\_SOO ] : PARSED\_FAB [ FABSV\_SOO ] = .FDL\$GL\_SWITCH;  
: 1154 1865 2  
: 1155 1866 2 [ FDLSC\_SOC ] : PARSED\_FAB [ FABSV\_SCF ] = .FDL\$GL\_SWITCH;  
: 1156 1867 2  
: 1157 1868 2 [ FDLSC\_SUPER ] : PARSED\_FAB [ FABSV\_SUP ] = .FDL\$GL\_SWITCH;

```
1158 1869 2  
1159 1870 2  
1160 1871 2  
1161 1872 2  
1162 1873 2  
1163 1874 2  
1164 1875 2  
1165 1876 2  
1166 1877 2  
1167 1878 2  
1168 1879 2  
1169 1880 2  
1170 1881 2  
1171 1882 2  
1172 1883 2  
1173 1884 1  
  
[ FDLSC_TEMPO ] : PARSED_FAB [ FABSV_TMD ] = .FDLSQL_SWITCH;  
[ FDLSC_TOC ] : PARSED_FAB [ FABSV_TEF ] = .FDLSQL_SWITCH;  
[ FDLSC_UFO ] : PARSED_FAB [ FABSV_UFO ] = .FDLSQL_SWITCH;  
[ FDLSC_WIN ] : PARSED_FAB [ FABSB_RTV ] = .FDLSQL_NUMBER;  
[ FDLSC_WRITEC ] : PARSED_FAB [ FABSV_WCK ] = .FDLSQL_SWITCH;  
  
TES:  
RETURN  
END;
```

VAX-11 FDL Utilities SET_FILE_P				16-Sep-1984 01:50:08				VAX-11 BLISS-32 V4.0-742 DISKSVMSMASTER:[FDL.SRC]FDLPARSE.B32;1				Page 35 (12)
04	A6	01	00000050	35	A6	58	90	000A9	78:	MOVBL	R8, 53(PARSED_FAB)	1784
04	A6	01	00000051	05	8F	57	D1	000AD	78:	CMPL	R7, #80	1787
05	A6	01	00000052	07	8F	69	F0	000B4	88:	BNEQ	8\$	1789
04	A6	01	00000054	03	8F	57	D1	000BC	88:	INSV	FDL\$GL SWITCH, #5, #1, 4(PARSED_FAB)	1791
04	A6	01	00000055	14	A6	69	F0	000B6	98:	CMPL	R7, #81	1796
04	A6	01	00000056	48	A6	57	D1	000CB	98:	BNEQ	9\$	1798
04	A6	01	00000057	3C	A6	69	F0	000D2	108:	INSV	FDL\$GL SWITCH, #7, #1, 5(PARSED_FAB)	1800
04	A6	01	00000058	00	8F	57	D1	000DA	108:	CMPL	R7, #82	1802
05	A6	01	00000059	02	8F	04	12	000E1	118:	BNEQ	10\$	1804
04	A6	01	0000005A	00	FB	6A	B0	000E3	118:	MOVW	FDL\$GL NUMBER, 20(PARSED_FAB)	1806
04	A6	01	0000005B	07	8F	57	D1	000E7	118:	CMPL	R7, #85	1808
05	A6	01	0000005C	03	8F	04	12	000EE	128:	BNEQ	12\$	1810
04	A6	01	0000005D	38	A6	6A	B0	000F0	128:	MOVW	FDL\$GL NUMBER, 72(PARSED_FAB)	1812
04	A6	01	0000005E	01	8F	57	D1	000F4	128:	CMPL	R7, #86	1814
04	A6	01	0000005F	50	FB	69	F0	0010A	138:	BNEQ	13\$	1816
04	A6	01	00000060	00	8F	57	D1	00110	138:	INSV	FDL\$GL SWITCH, #0, #1, 5(PARSED_FAB)	1819
04	A6	01	00000061	2C	A6	04	12	00108	148:	CMPL	R7, #88	1825
04	A6	01	00000062	60	8F	69	F0	00119	148:	BNEQ	15\$	1828
04	A6	01	00000063	34	A6	57	D1	00119	148:	INSV	FDL\$GL SWITCH, #2, #1, 5(PARSED_FAB)	1832
04	A6	01	00000064	00	8F	06	12	00126	158:	CMPL	R7, #89	1835
04	A6	01	00000065	00	FB	00	FB	00128	158:	BNEQ	16\$	1837
04	A6	01	00000066	00	8F	57	D1	0012F	158:	CALLS	#0, SET_PROT	1839
04	A6	01	00000067	50	FB	06	12	00136	168:	CMPL	R7, #90	1842
04	A6	01	00000068	07	8F	69	F0	00138	168:	BNEQ	17\$	1844
04	A6	01	00000069	02	8F	57	D1	0013E	168:	INSV	FDL\$GL SWITCH, #7, #1, 4(PARSED_FAB)	1846
04	A6	01	0000006A	00	FB	06	12	00145	178:	CMPL	R7, #91	1848
04	A6	01	0000006B	38	A6	69	F0	00147	178:	BNEQ	18\$	1850
04	A6	01	0000006C	03	8F	57	D1	0014D	178:	INSV	FDL\$GL SWITCH, #3, #1, 5(PARSED_FAB)	1852
04	A6	01	0000006D	00	FB	04	12	00154	188:	CMPL	R7, #92	1854
04	A6	01	0000006E	01	8F	69	F0	00156	188:	BNEQ	19\$	1856
04	A6	01	0000006F	50	FB	57	D1	0015A	188:	MOVL	FDL\$GL NUMBER, 56(PARSED_FAB)	1858
04	A6	01	00000070	00	8F	06	12	00161	198:	CMPL	R7, #93	1860
04	A6	01	00000071	01	FB	69	F0	00163	198:	BNEQ	20\$	1862
04	A6	01	00000072	50	8F	57	D1	00169	198:	INSV	FDL\$GL SWITCH, #1, #1, 4(PARSED_FAB)	1864
04	A6	01	00000073	50	FB	1F	12	00170	208:	CMPL	R7, #94	1866
04	A6	01	00000074	50	8F	68	3C	00172	208:	BNEQ	22\$	1868
04	A6	01	00000075	50	FB	16	13	00175	208:	MOVZWL	FDL\$AB_STRING, R0	1870
04	A6	01	00000076	50	8F	50	DD	00177	208:	BEQL	21\$	1872
04	A6	01	00000077	00	FB	01	FB	00179	208:	PUSHL	RO	1874
04	A6	01	00000078	2C	A6	50	DD	00180	208:	CALLS	#1, FDLSGET VM	1876
04	A6	01	00000079	50	FB	04	AB	00184	208:	MOVL	RO, 44(PARSED_FAB)	1878
04	A6	01	00000080	60	8F	68	28	00188	218:	MOVL	FDLSAB_STRING\$4, RO	1880
04	A6	01	00000081	34	A6	68	90	0018D	218:	MOVC3	FDLSAB_STRING, {RO}, @44(PARSED_FAB)	1882
04	A6	01	00000082	00	8F	57	D1	00191	218:	MOVB	FDLSAB_STRING, 52(PARSED_FAB)	1884
04	A6	01	00000083	00	FB	06	12	00198	228:	CMPL	R7, #95	1886
04	A6	01	00000084	50	8F	69	F0	0019A	228:	BNEQ	23\$	1888
04	A6	01	00000085	00	FB	57	D1	001A0	228:	INSV	FDL\$GL SWITCH, #0, #1, 6(PARSED_FAB)	1890
04	A6	01	00000086	1D	A6	08	12	001A7	228:	CMPL	R7, #96	1892
04	A6	01	00000087	00	FB	08	90	001A9	228:	BNEQ	24\$	1894
04	A6	01	00000088	00	FB	57	D1	001B1	228:	MOVB	FDL\$GL QUALIFIER, 29(PARSED_FAB)	1896
04	A6	01	00000089	50	8F	00	90	001A9	228:	CMPL	R7, #97	1898

07	A6	01	00000063	05	06	12	001B8	BNEQ	25\$		
			8F		69	F0	001BA	INSV	FDL\$GL_SWITCH, #5, #1, 7(PARSED_FAB)		
					57	D1	001C0	CMPL	R7 #99		
			00000000V	00	07	12	001C7	BNEQ	26\$		
			00000064	8F	00	FB	001C9	CALLS	#0, SET PROT		
					57	D1	001D0	CMPL	R7 #100		
					06	12	001D7	BNEQ	27\$		
05	A6	01	00000065	05	69	F0	001D9	INSV	FDL\$GL_SWITCH, #5, #1, 5(PARSED_FAB)		
			8F		57	D1	001DF	CMPL	R7 #101		
					07	12	001E6	BNEQ	28\$		
			00000000V	00	00	FB	001E8	CALLS	#0, SET PROT		
			00000066	8F	57	D1	001EF	CMPL	R7 #102		
					06	12	001F6	BNEQ	29\$		
06	A6	01	00000067	07	69	F0	001F8	INSV	FDL\$GL_SWITCH, #7, #1, 6(PARSED_FAB)		
			8F		57	D1	001FE	CMPL	R7 #103		
					24	12	00205	BNEQ	31\$		
			00000000'	00	00	D5	00207	TSTL	REVISION_XAB		
					11	12	0020D	BNEQ	30\$		
			00000000V	7E	1E	7D	0020F	MOVQ	#30, -(SP)		
			00		02	FB	00212	CALLS	#2, ALLOCATE_XAB		
			00000000'	00	50	D0	00219	MOVL	R0, REVISION_XAB		
				50	00000000'	00	DD	00220	REVISION_XAB, R0		
			08	A0	6A	B0	00227	MOVLW	FDL\$GL_NUMBER, 8(R0)		
			00000068	8F	57	D1	00228	CMPL	R7 #104		
					06	12	00232	BNEQ	32\$		
04	A6	01	00000069	06	69	F0	00234	INSV	FDL\$GL_SWITCH, #6, #1, 4(PARSED_FAB)		
			8F		57	D1	0023A	CMPL	R7 #105		
					06	12	00241	BNEQ	33\$		
05	A6	01	0000006A	06	69	F0	00243	INSV	FDL\$GL_SWITCH, #6, #1, 5(PARSED_FAB)		
			8F		57	D1	00249	CMPL	R7 #106		
					06	12	00250	BNEQ	34\$		
04	A6	01	0000006B	02	69	F0	00252	INSV	FDL\$GL_SWITCH, #2, #1, 4(PARSED_FAB)		
			8F		57	D1	00258	CMPL	R7 #107		
					06	12	0025F	BNEQ	35\$		
04	A6	01	0000006C	04	69	F0	00261	INSV	FDL\$GL_SWITCH, #4, #1, 4(PARSED_FAB)		
			8F		57	D1	00267	CMPL	R7 #108		
					06	12	0026E	BNEQ	36\$		
07	A6	01	0000006D	04	69	F0	00270	INSV	FDL\$GL_SWITCH, #4, #1, 7(PARSED_FAB)		
			8F		57	D1	00276	CMPL	R7 #109		
					06	12	0027D	BNEQ	37\$		
06	A6	01	0000006E	01	69	F0	0027F	INSV	FDL\$GL_SWITCH, #1, #1, 6(PARSED_FAB)		
			8F		57	D1	00285	CMPL	R7 #110		
					04	12	0028C	BNEQ	38\$		
			1C	A6	6A	90	0028E	MOVB	FDL\$GL_NUMBER, 28(PARSED_FAB)		
			0000006F	8F	57	D1	00292	CMPL	R7 #111		
					06	12	00299	BNEQ	39\$		
05	A6	01		01	69	F0	0029B	INSV	FDL\$GL_SWITCH, #1, #1, 5(PARSED_FAB)		
					04	002A1	39\$:	RET			

; Routine Size: 674 bytes, Routine Base: \_FDL\$CODE + 04C8

```
:1175 1885 1 XSBTTL 'SET KEY P'  
:1176 1886 1 ROUTINE SET_KEY_P : NOVALUE =  
:1177 1887 1 ++  
:1178 1888 1 Functional Description:  
:1179 1889 1  
:1180 1890 1 Fill in the blanks for the key xab  
:1181 1891 1  
:1182 1892 1 Calling Sequence:  
:1183 1893 1 set_key_p()  
:1184 1894 1  
:1185 1895 1 Input Parameters:  
:1186 1896 1 none  
:1187 1897 1  
:1188 1898 1 Implicit Inputs:  
:1189 1899 1  
:1190 1900 1 1901 1 fdI$secondary - Secondary code  
:1191 1902 1  
:1192 1903 1  
:1193 1904 1 Output Parameters:  
:1194 1905 1 none  
:1195 1906 1  
:1196 1907 1 Implicit Outputs:  
:1197 1908 1 none  
:1198 1909 1  
:1199 1910 1 Routine Value:  
:1200 1911 1 none  
:1201 1912 1  
:1202 1913 1 Routines Called:  
:1203 1914 1 allocate_xab  
:1204 1915 1  
:1205 1916 1  
:1206 1917 1 Side Effects:  
:1207 1918 1 none  
:1208 1919 1  
:1209 1920 1 --  
:1210 1921 1  
:1211 1922 2 BEGIN  
:1212 1923 2  
:1213 1924 2 ! Find out if there is a current xab if not then get one  
:1214 1925 2  
:1215 1926 2 IF .CURRENT_XAB EQ 0  
:1216 1927 2 THEN  
:1217 1928 3 BEGIN  
:1218 1929 3  
:1219 1930 3 ALLOCATE_XAB ( XABSC_KEY, .FDL$GL_PRINUM );  
:1220 1931 3  
:1221 1932 3 CURRENT_XAB [ XABSB_REF ] = .FDL$GL_PRINUM  
:1222 1933 3  
:1223 1934 3 END  
:1224 1935 2 ELSE  
:1225 1936 2  
:1226 1937 2 ! If the current xab is not the same type or number of what we want  
:1227 1938 2 ! then get a new one  
:1228 1939 2  
:1229 1940 2 IF ( .CURRENT_XAB [ XABSB_COD ] NEQ XABSC_KEY ) OR  
:1230 1941 2 ( .CURRENT_XAB [ XABSB_REF ] NEQ .FDL$GL_PRINUM )
```

```

1232      1942 2      THEN
1233      1943 3      BEGIN
1234      1944 3
1235      1945 3      ALLOCATE_XAB ( XABSC_KEY, .FDL$GL_PRINUM );
1236      1946 3
1237      1947 3      CURRENT_XAB [ XABSB_REF ] = .FDL$GL_PRINUM
1238      1948 3
1239      1949 2      END;
1240      1950 2
1241      1951 2      ! Set the key xab fields
1242      1952 2
1243      1953 2      CASE .FDL$GL_SECONDARY FROM FDLSC_CHANGE TO FDLSC_SEGTYP OF
1244      1954 2      SET
1245      1955 2      [ FDLSC_CHANGE ]: CURRENT_XAB [ XABSV_CHG ] = .FDL$GL_SWITCH;
1246      1956 2
1247      1957 2      [ FDLSC_DAREA ] : CURRENT_XAB [ XABSB_DAN ] = .FDL$GL_NUMBER;
1248      1958 2
1249      1959 2      [ FDLSC_DFILL ] : CURRENT_XAB [ XABSW_DFL ] = .FDL$GL_NUMBER;
1250      1960 2
1251      1961 2      [ FDLSC_DATKC ] : CURRENT_XAB [ XABSV_KEY_NCMPR ] = NOT .FDL$GL_SWITCH;
1252      1962 2
1253      1963 2      [ FDLSC_DATRC ] : CURRENT_XAB [ XABSV_DAT_NCMPR ] = NOT .FDL$GL_SWITCH;
1254      1964 2
1255      1965 2      [ FDLSC_DUPS ] : CURRENT_XAB [ XABSV_DUP ] = .FDL$GL_SWITCH;
1256      1966 2
1257      1967 2      [ FDLSC_IAREA ] : CURRENT_XAB [ XABSB_IAN ] = .FDL$GL_NUMBER;
1258      1968 2
1259      1969 2      [ FDLSC_IDXC ] : CURRENT_XAB [ XABSV_IDX_NCMPR ] = NOT .FDL$GL_SWITCH;
1260      1970 2
1261      1971 2      [ FDLSC_IFILL ] : CURRENT_XAB [ XABSW_IFL ] = .FDL$GL_NUMBER;
1262      1972 2
1263      1973 3      [ FDLSC_KYNAME ]: BEGIN
1264      1974 3      CURRENT_XAB [ XABSL_KNM ] = FDLS$GET VM ( 32 );
1265      1975 3      CHSCOPY( .FDL$AB_STRING [ DSCSW_LENGTH ],
1266      1976 3      .FDL$AB_STRING [ DSCSA_POINTER ],
1267      1977 3      SPACE,32,
1268      1978 3      .CURRENT_XAB [ XABSL_KNM ] )
1269      1979 2      END;
1270      1980 2
1271      1981 2      [ FDLSC_LAREA ] : CURRENT_XAB [ XABSB_LAN ] = .FDL$GL_NUMBER;
1272      1982 2
1273      1983 2      [ FDLSC_NULL ] : CURRENT_XAB [ XABSV_NUL ] = .FDL$GL_SWITCH;
1274      1984 2
1275      1985 2      [ FDLSC_NULLVAL]: CURRENT_XAB [ XABSB_NUL ] = .FDL$GL_QUALIFIER;
1276      1986 2
1277      1987 2      [ FDLSC_PROL ] : IF .CURRENT_XAB [ XABSB_REF ] EQLU 0
1278      1988 2      THEN
1279      1989 2      CURRENT_XAB [ XABSB_PROLOG ] = .FDL$GL_NUMBER;
1280      1990 2
1281      1991 2      [ FDLSC_SEGLEN ]: CASE .FDL$GL_SECNUM FROM 0 TO 7 OF
1282      1992 2      SET
1283      1993 2      [ 0 ] : CURRENT_XAB [ XABSB_SIZ0 ] = .FDL$GL_NUMBER;
1284      1994 2      [ 1 ] : CURRENT_XAB [ XABSB_SIZ1 ] = .FDL$GL_NUMBER;
1285      1995 2      [ 2 ] : CURRENT_XAB [ XABSB_SIZ2 ] = .FDL$GL_NUMBER;
1286      1996 2      [ 3 ] : CURRENT_XAB [ XABSB_SIZ3 ] = .FDL$GL_NUMBER;
1287      1997 2      [ 4 ] : CURRENT_XAB [ XABSB_SIZ4 ] = .FDL$GL_NUMBER;
1288      1998 2      [ 5 ] : CURRENT_XAB [ XABSB_SIZ5 ] = .FDL$GL_NUMBER;

```

```

1289      1999 2
1290      2000 2
1291      2001 2
1292      2002 2
1293      2003 2
1294      2004 2
1295      2005 2
1296      2006 2
1297      2007 2
1298      2008 2
1299      2009 2
1300      2010 2
1301      2011 2
1302      2012 2
1303      2013 2
1304      2014 2
1305      2015 2
1306      2016 2
1307      2017 2
1308      2018 2
1309      2019 2
1310      2020 2
1311      2021 2
1312      2022 2
1313      2023 2
1314      2024 2
1315      2025 2
1316      2026 2
1317      2027 2
1318      2028 2
1319      2029 2
1320      2030 2
1321      2031 2
1322      2032 2
1323      2033 2
1324      2034 1

[ 6 ] : CURRENT_XAB [ XAB$B_SIZE ] = .FDL$GL_NUMBER;
[ 7 ] : CURRENT_XAB [ XAB$B_SIZE ] = .FDL$GL_NUMBER;
TES;

[ FDL$C_SEGPOS ]: CASE .FDL$GL_SECNUM FROM 0 TO 7 OF
SET
[ 0 ] : CURRENT_XAB [ XAB$W_POS0 ] = .FDL$GL_NUMBER;
[ 1 ] : CURRENT_XAB [ XAB$W_POS1 ] = .FDL$GL_NUMBER;
[ 2 ] : CURRENT_XAB [ XAB$W_POS2 ] = .FDL$GL_NUMBER;
[ 3 ] : CURRENT_XAB [ XAB$W_POS3 ] = .FDL$GL_NUMBER;
[ 4 ] : CURRENT_XAB [ XAB$W_POS4 ] = .FDL$GL_NUMBER;
[ 5 ] : CURRENT_XAB [ XAB$W_POS5 ] = .FDL$GL_NUMBER;
[ 6 ] : CURRENT_XAB [ XAB$W_POS6 ] = .FDL$GL_NUMBER;
[ 7 ] : CURRENT_XAB [ XAB$W_POS7 ] = .FDL$GL_NUMBER;
TES;

[ FDL$C_SEGTYP ]: CASE .FDL$GL_SECNUM FROM 0 TO 7 OF
SET
[ 0 ] : BEGIN
CURRENT_XAB [ XAB$B_DTP ] = .FDL$GL_QUALIFIER;
CURRENT_XAB [ XAB$B_TYP0 ] = .FDL$GL_QUALIFIER
END;
[ 1 ] : CURRENT_XAB [ XAB$B_TYP1 ] = .FDL$GL_QUALIFIER;
[ 2 ] : CURRENT_XAB [ XAB$B_TYP2 ] = .FDL$GL_QUALIFIER;
[ 3 ] : CURRENT_XAB [ XAB$B_TYP3 ] = .FDL$GL_QUALIFIER;
[ 4 ] : CURRENT_XAB [ XAB$B_TYP4 ] = .FDL$GL_QUALIFIER;
[ 5 ] : CURRENT_XAB [ XAB$B_TYP5 ] = .FDL$GL_QUALIFIER;
[ 6 ] : CURRENT_XAB [ XAB$B_TYP6 ] = .FDL$GL_QUALIFIER;
[ 7 ] : CURRENT_XAB [ XAB$B_TYP7 ] = .FDL$GL_QUALIFIER;
TES;

TES;
RETURN
END;

```

69 17 A2

OFFC 00000 SET_KEY_P:						
5B	00000000G	00	9E 00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 1886
5A	00000000G	00	9E 00009	MOVAB	FDL\$GL_QUALIFIER, R11	
59	00000000G	00	9E 00010	MOVAB	FDL\$GL_SECNUM, R10	
58	00000000	00	9E 00017	MOVAB	FDL\$GL_PRINUM, R9	
57	00000000G	00	9E 0001E	MOVAB	CURRENT_XAB, R8	
56	00000000G	00	9E 00025	MOVAB	FDL\$GL_SWITCH, R7	
52		68	D0 0002C	MOVL	FDL\$GL_NUMBER, R6	
		00	13 0002F	BEQL	CURRENT_XAB, R2	
15		62	91 00031	CMPB	1\$	
		08	12 00034	BNEQ	(R2), #21	
08		00	ED 00036	CMPZV	1\$	
		12	13 0003C	BEQL	#0, #8, 23(R2), FDL\$GL_PRINUM	
		69	DD 0003E 18:	PUSHL	2\$	
		15	DD 00040	PUSHL	FDL\$GL_PRINUM	
					#21	

0033	00000000V	00	02	FB	00042	CALLS	#2, ALLOCATE_XAB	1947	
		17	A0	68	00049	MOVL	CURRENT_XAB, R0		
			52	69	0004C	MOVB	FDL\$GL_PRINUM, 23(R0)	1955	
	10 00000077	8F	00000000G	68	00050	28:	CURRENT_XAB, R2		
	002E	0029	0022	CF	00053	38:	FDL\$GL_SECONDARY, #119, #16	1953	
	004E	0047	003D		00067	CASEL	48-38,-		
	0062	0085	005D		0006F	.WORD	58-38,-		
	00A1	0096	0091		00077		68-38,-		
			011F		0007F		78-38,-		
							88-38,-		
							98-38,-		
							108-38,-		
							118-38,-		
							128-38,-		
							148-38,-		
							138-38,-		
							158-38,-		
							168-38,-		
							178-38,-		
							198-38,-		
							298-38,-		
							398-38		
A2	01	01	67	F0	00081	48:	INSV	FDL\$GL_SWITCH, #1, #1, 18(R2)	1955
		0A	A2	66	90	00088	58:	RET	
				04	0008C		MOVB	FDL\$GL_NUMBER, 10(R2)	1957
		1C	A2	66	B0	0008D	68:	RET	
				04	00091		MOVW	FDL\$GL_NUMBER, 28(R2)	1959
A2	01	50	67	D2	00092	78:	MCOML	FDL\$GL_SWITCH, R0	1961
		06	50	50	F0	00095	INSV	R0, #6, #1, 18(R2)	
				04	00098		RET		
A2	01	50	67	D2	0009C	88:	MCOML	FDL\$GL_SWITCH, R0	1963
		07	50	50	F0	0009F	INSV	R0, #7, #1, 18(R2)	
A2	01	00	67	F0	000A6	98:	RET		
		04	000AC		04	000A5	INSV	FDL\$GL_SWITCH, #0, #1, 18(R2)	1965
	08	A2	66	90	000AD	108:	MOVB	FDL\$GL_NUMBER, 8(R2)	1967
A2	01	50	66	04	000B1		RET		
	03	03	67	D2	000B2	118:	MCOML	FDL\$GL_SWITCH, R0	1969
			50	F0	000B5		INSV	R0, #3, #1, 18(R2)	
	1A	A2	66	B0	000BC	128:	RET		
			04	000C0			MOVW	FDL\$GL_NUMBER, 26(R2)	1971
			20	DD	000C1	138:	RET		
	00000000V	00	01	FB	000C3		PUSHL	#32	1974
	38	A2	50	D0	000CA		CALLS	#1, FDL\$GET_VM	
		51 00000000G	00	D0	000CE		MOVL	R0, 56(R2)	
		50	68	D0	000D5		MOVL	FDL\$AB_STRING+4, R1	1976
20	20	61 00000000G	00	2C	000D8		MOVL	CURRENT_XAB, R0	1978
		38	B0		000E1		MOVCS	FDL\$AB_STRING, (R1), #32, #32, 256(R0)	
				04	000E3	RET			
				04	000E8	RET			
	09	A2	66	90	000E4	148:	MOVB	FDL\$GL_NUMBER, 9(R2)	1981
				04	000E9		RET		
A2	01	02	67	F0	000E9	158:	INSV	FDL\$GL_SWITCH, #2, #1, 18(R2)	1983
	15	A2	68	90	000F0	168:	MOVB	FDL\$GL_QUALIFIER, 21(R2)	1985
				04	000F4		RET		

001F 0033	07 001A 002E	48 50 00 0015 0029	17 66 66 6A 0010 0024	02 95 000F5 17\$: TSTB 23(R2) 01 15 000FB BEQL 18\$	1987 1989 1987 1993 1991 1993 1994 1995 1996 1997 1998 1999 2000 1991 2005 2003
				66 90 000FB 18\$: MOV8 FDLSGL_NUMBER, 72(R2)	
				04 000FA RET	
				66 D0 00100 19\$: MOVL FDL\$GL_NUMBER, R0	
				6A CF 00103 19\$: CASEL FDL\$GL_SECNUM, #0, #7	
				00107 20\$: .WORD 21\$-20\$,-	
				0010F 20\$: .WORD 22\$-20\$,-	
				20\$: .WORD 23\$-20\$,-	
				20\$: .WORD 24\$-20\$,-	
				20\$: .WORD 25\$-20\$,-	
001F 0033	07 001A 002E	2E 2F 30 31 32 33 34 35 50 00 0015 0029	50 90 00117 21\$: MOV8 R0, 46(R2) 50 90 0011B RET 50 90 0011C 22\$: MOV8 R0, 47(R2) 50 90 00120 RET 50 90 00121 23\$: MOV8 R0, 48(R2) 50 90 00125 RET 50 90 00126 24\$: MOV8 R0, 49(R2) 50 90 0012A RET 50 90 0012B 25\$: MOV8 R0, 50(R2) 50 90 00130 26\$: MOV8 R0, 51(R2) 50 90 00134 RET 50 90 00135 27\$: MOV8 R0, 52(R2) 50 90 0013A 28\$: MOV8 R0, 53(R2) 66 D0 0013F 29\$: MOVL FDL\$GL_NUMBER, R0 6A CF 00142 CASEL FDL\$GL_SECNUM, #0, #7	1993 1994 1995 1996 1997 1998 1999 2000 1991 2005 2003	
				00142 30\$: .WORD 31\$-30\$,-	
				0014E 30\$: .WORD 32\$-30\$,-	
				30\$: .WORD 33\$-30\$,-	
				30\$: .WORD 34\$-30\$,-	
				30\$: .WORD 35\$-30\$,-	
				30\$: .WORD 36\$-30\$,-	
				30\$: .WORD 37\$-30\$,-	
				30\$: .WORD 38\$-30\$,-	
				30\$: .WORD R0, 30(R2)	
001F 0033	07 001A 002E	1E 20 22 24 26 28 2A 2C	50 B0 00156 31\$: MOVW R0, 32(R2) 50 B0 0015A RET 50 B0 0015B 32\$: MOVW R0, 34(R2) 50 B0 00160 RET 50 B0 00164 33\$: MOVW R0, 36(R2) 50 B0 00165 RET 50 B0 0016A 35\$: MOVW R0, 38(R2) 50 B0 0016E RET 50 B0 0016F 36\$: MOVW R0, 40(R2) 50 B0 00173 RET 50 B0 00174 37\$: MOVW R0, 42(R2) 50 B0 00178 RET 50 B0 00179 38\$: MOVW R0, 44(R2) 04 0017D RET	2005 2006 2007 2008 2009 2010 2011 2012 2003	
				0017D 38\$: RET	

			68	D0	0017E	398:	MOVL	FDLSGL_QUALIFIER, R0	: 2018		
			6A	CF	00181		CASEL	FDLSGL_SECNUM, #0, #7	: 2015		
0023	07	50	0010	00185	408:	.WORD	418-40\$,-				
0037	001E	00	0028	0018D			428-40\$,-				
							438-40\$,-				
							448-40\$,-				
							458-40\$,-				
							468-40\$,-				
							478-40\$,-				
							488-40\$				
			13	A2	50	90	00195	418:	MOVBL	RO, 19(R2)	: 2018
			40	A2	50	90	00199		MOVBL	RO, 64(R2)	: 2019
					04	0019D		RET			
			41	A2	50	90	0019E	428:	MOVBL	RO, 65(R2)	: 2021
					04	001A2		RET			
			42	A2	50	90	001A3	438:	MOVBL	RO, 66(R2)	: 2022
					04	001A7		RET			
			43	A2	50	90	001A8	448:	MOVBL	RO, 67(R2)	: 2023
					04	001AC		RET			
			44	A2	50	90	001AD	458:	MOVBL	RO, 68(R2)	: 2024
					04	001B1		RET			
			45	A2	50	90	001B2	468:	MOVBL	RO, 69(R2)	: 2025
					04	001B6		RET			
			46	A2	50	90	001B7	478:	MOVBL	RO, 70(R2)	: 2026
					04	001BB		RET			
			47	A2	50	90	001BC	488:	MOVBL	RO, 71(R2)	: 2027
					04	001C0		RET			: 2034

: Routine Size: 449 bytes. Routine Base: \_FDLSCODE + 076A

1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382

2035 1 XSBTTL 'SET\_RECORD\_P'  
2036 1 ROUTINE SET\_RECORD\_P : NOVALUE =  
2037 1 ++  
2038 1 Functional Description:  
2039 1 Fill in the blanks for the fab fields concerning the record  
2040 1 Calling Sequence:  
2041 1 set\_record\_p()  
2042 1 Input Parameters:  
2043 1 none  
2044 1 Implicit Inputs:  
2045 1 fdlssecondary - Secondary code  
2046 1 Output Parameters:  
2047 1 none  
2048 1 Implicit Outputs:  
2049 1 none  
2050 1 Routine Value:  
2051 1 none  
2052 1 Routines Called:  
2053 1 none  
2054 1 Side Effects:  
2055 1 none  
2056 1 --  
2057 1 BEGIN  
2058 1 REGISTER  
2059 1 PARSED\_FAB : REF BLOCK [ ,BYTE ];  
2060 1 PARSED\_FAB = .FDLSAB\_PARSED\_FAB;  
2061 1 ! Set em up  
2062 1 CASE .FDL\$GL\_SECONDARY FROM FDLSIZE\_TO\_FDLSC\_SIZE\_OF  
2063 1 SET [ FDLSIZE\_BLKSPN ]: PARSED\_FAB [ FABSV\_BLK ] = NOT .FDL\$GL\_SWITCH;  
2064 1 [ FDLSIZE\_CACTRL ]: CASE .FDL\$GL\_QUALIFIER FROM FDLSIZE\_NONE\_TO\_FDLSC\_PRINT\_OF  
2065 1 SET ! We must clear the other flags while setting the one  
2066 1 ! we want (without clearing BLK if set)  
2067 1 [ FDLSIZE\_NONE ] : PARSED\_FAB [ FABSB\_RAT ] =  
2068 1 .PARSED\_FAB [ FABSB\_RAT ] AND  
2069 1 FABSM\_BLK;  
2070 1  
2071 2  
2072 2  
2073 2  
2074 2  
2075 2  
2076 2  
2077 2  
2078 2  
2079 2  
2080 2  
2081 2  
2082 2  
2083 2  
2084 2  
2085 2  
2086 2  
2087 2  
2088 2  
2089 2  
2090 2  
2091 2



FDLPARSE  
V04-000

VAX-11 FDL Utilities  
SET\_RECORD\_P

H 8  
16-Sep-1984 01:50:08 VAX-11 BLISS-32 V4.0-742  
14-Sep-1984 12:31:19 DISK\$VMSMASTER:[FDL.SRC]FDLPARSE.B32;1

Page 45  
(14)

61	50 FFFFFFF7	8F	CA 00071	BICL2	#-9, R0	:	2100
	50	04	89 00078	BISB3	#4, R0, (R1)	:	2084
			04 0007C	RET		:	2103
3F	A0	63	90 0007D 98:	MOV B	FDLSGL_NUMBER, 63(PARSED_FAB)	:	2105
			04 00081	RET		:	2107
1F	A0	62	90 00082 108:	MOV B	FDLSGL_QUALIFIER, 31(PARSED_FAB)	:	2112
			04 00086	RET		:	
36	A0	63	B0 00087 118:	MOV W	FDLSGL_NUMBER, 54(PARSED_FAB)	:	
			04 0008B	RET		:	

; Routine Size: 140 bytes, Routine Base: \_FDLSCODE + 092B

```
: 1405
: 1406
: 1407
: 1408
: 1409
: 1410
: 1411
: 1412
: 1413
: 1414
: 1415
: 1416
: 1417
: 1418
: 1419
: 1420
: 1421
: 1422
: 1423
: 1424
: 1425
: 1426
: 1427
: 1428
: 1429
: 1430
: 1431
: 1432
: 1433
: 1434
: 1435
: 1436
: 1437
: 1438
: 1439
: 1440
: 1441
: 1442
: 1443
: 1444
: 1445
: 1446
: 1447
: 1448
: 1449
: 1450
: 1451
: 1452
: 1453
: 1454
: 1455
: 1456
: 1457
: 1458
: 1459
: 1460
: 1461
2113 1 XSBTTL 'SET_ACCESS_P'
2114 1 ROUTINE SET_ACCESS_P : NOVALUE =
2115 1 ++
2116 1
2117 1 Functional Description:
2118 1     Fill in the blanks for the fab fields concerning access mode
2119 1
2120 1 Calling Sequence:
2121 1     set_access_p()
2122 1
2123 1 Input Parameters:
2124 1     none
2125 1
2126 1 Implicit Inputs:
2127 1
2128 1     fdl$secondary - Secondary code
2129 1
2130 1 Output Parameters:
2131 1     none
2132 1
2133 1 Implicit Outputs:
2134 1     none
2135 1
2136 1 Routine Value:
2137 1     none
2138 1
2139 1 Routines Called:
2140 1     none
2141 1
2142 1 Side Effects:
2143 1     none
2144 1
2145 1
2146 1
2147 1 --+
2148 1
2149 2 BEGIN
2150 2
2151 2 REGISTER
2152 2     PARSED_FAB : REF BLOCK [ ,BYTE ];
2153 2
2154 2     PARSED_FAB = .FDL$AB_PARSED_FAB;
2155 2
2156 2
2157 2
2158 2
2159 2 CASE .FDL$GL_SECONDARY FROM FDLSC_FACBIO TO FDLSC_FACUPD OF
2160 2     SET
2161 2         [ FDLSC_FACBIO ] : PARSED_FAB [ FAB$V_BIO ] = .FDL$GL_SWITCH;
2162 2         [ FDLSC_FACDEL ] : PARSED_FAB [ FAB$V_DEL ] = .FDL$GL_SWITCH;
2163 2         [ FDLSC_FACGET ] : PARSED_FAB [ FAB$V_GET ] = .FDL$GL_SWITCH;
2164 2         [ FDLSC_FACPUT ] : PARSED_FAB [ FAB$V_PUT ] = .FDL$GL_SWITCH;
2165 2         [ FDLSC_FACBRO ] : PARSED_FAB [ FAB$V_BRO ] = .FDL$GL_SWITCH;
```

1462 2170 2 [ FDLSC\_FACTRN ] : PARSED\_FAB [ FABSV\_TRN ] = .FDLSQL\_SWITCH;  
1463 2171 2  
1464 2172 2 [ FDLSC\_FACUPD ] : PARSED\_FAB [ FABSV\_UPD ] = .FDLSQL\_SWITCH;  
1465 2173 2  
1466 2174 2 TES:  
1467 2175 2 RETURN  
1468 2176 2  
1469 2177 1 END;

0000 00000 SET_ACCESS P:									
								WORD	
		50 00000000G	00	DO 00002			MOVL	Save nothing	2114
		51 16 A0	9E 00009				MOVAB	FDLSAB_PARSED_FAB, PARSED_FAB	2154
		50 00000000G	00	DO 0000D			MOVL	22(PARSED_FAB), R1	2160
		01 00000000G	00	CF 00014			CASEL	FDL_SGL_SWITCH, R0	
		0014 000E	0001C	18:			WORD	FDL_SGL_SECONDARY, #1, #6	2158
		0032 002C	0026	00024				2\$-1\$,-	
								3\$-1\$,-	
								4\$-1\$,-	
								5\$-1\$,-	
								6\$-1\$,-	
								7\$-1\$,-	
								8\$-1\$,-	
0020	06								
	001A								
	0032								
61	01	05	50 F0 0002A	2\$:	INSV		RO, #5, #1, (R1)		2160
61	01	02	50 F0 00030	3\$:	INSV	RET	RO, #2, #1, (R1)		2162
61	01	01	50 F0 00036	4\$:	INSV	RET	RO, #1, #1, (R1)		2164
61	01	00	50 F0 0003C	5\$:	INSV	RET	RO, #0, #1, (R1)		2166
61	01	06	50 F0 00042	6\$:	INSV	RET	RO, #6, #1, (R1)		2168
61	01	04	50 F0 00048	7\$:	INSV	RET	RO, #4, #1, (R1)		2170
61	01	03	50 F0 0004E	8\$:	INSV	RET	RO, #3, #1, (R1)		2172
			04 00053						2177

: Routine Size: 84 bytes, Routine Base: \_FDL\$CODE + 09B7

```
: 1471      2178 1 %SBTTL 'SET_SHARING_P'  
: 1472      2179 1 ROUTINE SET_SHARING_P : NOVALUE =  
: 1473      2180 1 ++  
: 1474      2181 1 Functional Description:  
: 1475      2182 1     Fill in the blanks for the fab fields concerning sharing  
: 1476      2183 1 Calling Sequence:  
: 1477      2184 1     set_sharing_p()  
: 1478      2185 1 Input Parameters:  
: 1479      2186 1     none  
: 1480      2187 1 Implicit Inputs:  
: 1481      2188 1     fdL$secondary - Secondary code  
: 1482      2189 1 Output Parameters:  
: 1483      2190 1     none  
: 1484      2191 1 Implicit Outputs:  
: 1485      2192 1     none  
: 1486      2193 1 Routine Value:  
: 1487      2194 1     none  
: 1488      2195 1 Routines Called:  
: 1489      2196 1     none  
: 1490      2197 1 Side Effects:  
: 1491      2198 1     none  
: 1492      2199 1 --  
: 1493      2200 1     BEGIN  
: 1494      2201 1     REGISTER  
: 1495      2202 1       PARSED_FAB : REF BLOCK [ ,BYTE ];  
: 1496      2203 1     PARSED_FAB = .FDL$AB_PARSED_FAB;  
: 1497      2204 1     ! Set em up  
: 1498      2205 1     CASE .FDL$GL_SECONDARY FROM FDL$C_SHRDEL TO FDL$C_SHRUP1 OF  
: 1499      2206 1     SET  
: 1500      2207 1       [ FDL$C_SHRDEL ] : PARSED_FAB [ FAB$V_SHRDEL ] = .FDL$GL_SWITCH;  
: 1501      2208 1       [ FDL$C_SHRGET ] : PARSED_FAB [ FAB$V_SHRGET ] = .FDL$GL_SWITCH;  
: 1502      2209 1       [ FDL$C_SHRMSE ] : PARSED_FAB [ FAB$V_MSE ] = .FDL$GL_SWITCH;  
: 1503      2210 1       [ FDL$C_SHRNIL ] : PARSED_FAB [ FAB$V_NIL ] = .FDL$GL_SWITCH;  
: 1504      2211 1       [ FDL$C_SHRPUT ] : PARSED_FAB [ FAB$V_SHRPUT ] = .FDL$GL_SWITCH;  
: 1505      2212 1  
: 1506      2213 1  
: 1507      2214 2  
: 1508      2215 2  
: 1509      2216 2  
: 1510      2217 2  
: 1511      2218 2  
: 1512      2219 2  
: 1513      2220 2  
: 1514      2221 2  
: 1515      2222 2  
: 1516      2223 2  
: 1517      2224 2  
: 1518      2225 2  
: 1519      2226 2  
: 1520      2227 2  
: 1521      2228 2  
: 1522      2229 2  
: 1523      2230 2  
: 1524      2231 2  
: 1525      2232 2  
: 1526      2233 2  
: 1527      2234 2
```

```

: 1528 2235 2 [ FDLSC_SHRUPD ] : PARSED_FAB [ FAB$V_SHRUPD ] = .FDL$GL_SWITCH;
: 1529 2236 2 [ FDLSC_SHRUPI ] : PARSED_FAB [ FAB$V_UPI ] = .FDL$GL_SWITCH;
: 1530 2237 2 TES:
: 1531 2238 2
: 1532 2239 2 RETURN
: 1533 2240 2
: 1534 2241 2 END:
: 1535 2242 1

```

0000 00000 SET_SHARING_P:									
0020	06 0000008D	001A	50 0000000G	00 D0 00002	.WORD	Save nothing	2179		
			51 17	A0 9E 00009	MOVL	FDL\$AB_PARSED_FAB, PARSED_FAB	2219		
			50 0000000G	00 D0 0000D	MOVAB	23(PARSED_FAB), R1	2225		
			8F 0000000G	00 CF 00014	MOVL	FDL\$GL_SWITCH, R0			
				000E 00020	CASEL	FDL\$GL_SECONDARY, #141, #6	2223		
				0026 00028	.WORD	2\$-1\$,-			
						3\$-1\$,-			
						4\$-1\$,-			
61	01	02	50 F0 0002E	2\$: INSV	R0, #2, #1, (R1)	2225			
61	01	01	50 04 00033	RET					
61	01	04	50 F0 00034	3\$: INSV	R0, #1, #1, (R1)	2227			
61	01	04	50 04 00039	RET					
61	01	05	50 F0 0003A	4\$: INSV	R0, #4, #1, (R1)	2229			
61	01	05	50 04 0003F	RET					
61	01	00	50 F0 00040	5\$: INSV	R0, #5, #1, (R1)	2231			
61	01	00	50 04 00045	RET					
61	01	03	50 F0 00046	6\$: INSV	R0, #0, #1, (R1)	2233			
61	01	03	50 04 0004B	RET					
61	01	06	50 F0 0004C	7\$: INSV	R0, #3, #1, (R1)	2235			
61	01	06	50 04 00051	RET					
			50 F0 00052	8\$: INSV	R0, #6, #1, (R1)	2237			
			04 00057	RET		2242			

; Routine Size: 88 bytes. Routine Base: \_FDLSCODE + 0AOB

```
: 1537 2243 1 XSBTTL 'SET_CONNECT_P'  
. 1538 2244 1 ROUTINE SET_CONNECT_P : NOVALUE =  
. 1539 2245 1 ++  
. 1540 2246 1 Functional Description:  
. 1541 2247 1 Fill in the blanks for the Rab fields  
. 1542 2248 1 Calling Sequence:  
. 1543 2249 1 set_connect_p()  
. 1544 2250 1 Input Parameters:  
. 1545 2251 1 none  
. 1546 2252 1 Implicit Inputs:  
. 1547 2253 1 fdI$secondary - Secondary code  
. 1548 2254 1 Output Parameters:  
. 1549 2255 1 none  
. 1550 2256 1 Implicit Outputs:  
. 1551 2257 1 none  
. 1552 2258 1 Routine Value:  
. 1553 2259 1 none  
. 1554 2260 1 Routines Called:  
. 1555 2261 1 none  
. 1556 2262 1 Side Effects:  
. 1557 2263 1 none  
. 1558 2264 1 --  
. 1559 2265 1 BEGIN  
. 1560 2266 1 REGISTER  
. 1561 2267 1 PARSED_RAB : REF BLOCK [ ,BYTE ];  
. 1562 2268 1  
. 1563 2269 1 PARSED_RAB = .FDL$AB_PARSED_RAB;  
. 1564 2270 1  
. 1565 2271 1 ! Set em up  
. 1566 2272 1 CASE .FDL$GL_SECONDARY FROM FDL$C_ASY TO FDL$C_WBH OF  
. 1567 2273 1 SET  
. 1568 2274 1 [ FDL$C_ASY ] : PARSED_RAB [ RAB$V_ASY ] = .FDL$GL_SWITCH;  
. 1569 2275 1 [ FDL$C_BIO ] : PARSED_RAB [ RAB$V_BIO ] = .FDL$GL_SWITCH;  
. 1570 2276 1 [ FDL$C_BUCODE ] : PARSED_RAB [ RAB$L_BKT ] = .FDL$GL_NUMBER;  
. 1571 2277 1 [ FDL$C_RCTX ] : PARSED_RAB [ RAB$L_CTX ] = .FDL$GL_NUMBER;  
. 1572 2278 1 [ FDL$C_EOF ] : PARSED_RAB [ RAB$V_EOF ] = .FDL$GL_SWITCH;  
. 1573 2279 2  
. 1574 2280 2  
. 1575 2281 2  
. 1576 2282 2  
. 1577 2283 2  
. 1578 2284 2  
. 1579 2285 2  
. 1580 2286 2  
. 1581 2287 2  
. 1582 2288 2  
. 1583 2289 2  
. 1584 2290 2  
. 1585 2291 2  
. 1586 2292 2  
. 1587 2293 2  
. 1588 2294 2  
. 1589 2295 2  
. 1590 2296 2  
. 1591 2297 2  
. 1592 2298 2  
. 1593 2299 2
```

: 1594 2300 2 [ FDLSC\_FLOA ] : PARSED\_RAB [ RAB\$V\_LOA ] = .FDL\$GL\_SWITCH;  
: 1595 2301 2 [ FDLSC\_FDEL ] : PARSED\_RAB [ RAB\$V\_FDL ] = .FDL\$GL\_SWITCH;  
: 1596 2302 2 [ FDLSC\_KGE ] : PARSED\_RAB [ RAB\$V\_KGE ] = .FDL\$GL\_SWITCH;  
: 1597 2303 2 [ FDLSC\_KGT ] : PARSED\_RAB [ RAB\$V\_KGT ] = .FDL\$GL\_SWITCH;  
: 1598 2304 2 [ FDLSC\_KLIM ] : PARSED\_RAB [ RAB\$V\_LIM ] = .FDL\$GL\_SWITCH;  
: 1599 2305 2 [ FDLSC\_KRF ] : PARSED\_RAB [ RAB\$B\_KRF ] = .FDL\$GL\_NUMBER;  
: 1600 2306 2 [ FDLSC\_LOCMODE ] : PARSED\_RAB [ RAB\$V\_LOC ] = .FDL\$GL\_SWITCH;  
: 1601 2307 2 [ FDLSC\_REA ] : PARSED\_RAB [ RAB\$V\_REA ] = .FDL\$GL\_SWITCH;  
: 1602 2308 2 [ FDLSC\_RLK ] : PARSED\_RAB [ RAB\$V\_RLK ] = .FDL\$GL\_SWITCH;  
: 1603 2309 2 [ FDLSC\_ULK ] : PARSED\_RAB [ RAB\$V\_ULK ] = .FDL\$GL\_SWITCH;  
: 1604 2310 2 [ FDLSC\_MBC ] : PARSED\_RAB [ RAB\$B\_MBC ] = .FDL\$GL\_NUMBER;  
: 1605 2311 2 [ FDLSC\_MBFR ] : PARSED\_RAB [ RAB\$B\_MBFR ] = .FDL\$GL\_NUMBER;  
: 1606 2312 2 [ FDLSC\_NLK ] : PARSED\_RAB [ RAB\$V\_NLK ] = .FDL\$GL\_SWITCH;  
: 1607 2313 2 [ FDLSC\_NXR ] : PARSED\_RAB [ RAB\$V\_NXR ] = .FDL\$GL\_SWITCH;  
: 1608 2314 2 [ FDLSC\_RAH ] : PARSED\_RAB [ RAB\$V\_RAH ] = .FDL\$GL\_SWITCH;  
: 1609 2315 2 [ FDLSC\_RRL ] : PARSED\_RAB [ RAB\$V\_RRL ] = .FDL\$GL\_SWITCH;  
: 1610 2316 2 [ FDLSC\_TMO ] : PARSED\_RAB [ RAB\$B\_TMO ] = .FDL\$GL\_NUMBER;  
: 1611 2317 2 [ FDLSC\_TMENB ] : PARSED\_RAB [ RAB\$V\_TMO ] = .FDL\$GL\_SWITCH;  
: 1612 2318 2 [ FDLSC\_TPT ] : PARSED\_RAB [ RAB\$V\_TPT ] = .FDL\$GL\_SWITCH;  
: 1613 2319 2 [ FDLSC\_TTCCO ] : PARSED\_RAB [ RAB\$V\_CCO ] = .FDL\$GL\_SWITCH;  
: 1614 2320 2 [ FDLSC\_TTCVT ] : PARSED\_RAB [ RAB\$V\_CVT ] = .FDL\$GL\_SWITCH;  
: 1615 2321 2 [ FDLSC\_TTPMT ] : PARSED\_RAB [ RAB\$V\_PMT ] = .FDL\$GL\_SWITCH;  
: 1616 2322 2 [ FDLSC\_TTPTA ] : PARSED\_RAB [ RAB\$V\_PTA ] = .FDL\$GL\_SWITCH;  
: 1617 2323 2 [ FDLSC\_TTRNE ] : PARSED\_RAB [ RAB\$V\_RNE ] = .FDL\$GL\_SWITCH;  
: 1618 2324 2 [ FDLSC\_TTRNF ] : PARSED\_RAB [ RAB\$V\_RNF ] = .FDL\$GL\_SWITCH;  
: 1619 2325 2 [ FDLSC\_UIF ] : PARSED\_RAB [ RAB\$V\_UIF ] = .FDL\$GL\_SWITCH;  
: 1620 2326 2 [ FDLSC\_WAT ] : PARSED\_RAB [ RAB\$V\_WAT ] = .FDL\$GL\_SWITCH;  
: 1621 2327 2 [ FDLSC\_WBH ] : PARSED\_RAB [ RAB\$V\_WBH ] = .FDL\$GL\_SWITCH;  
: 1622 2328 2  
: 1623 2329 2  
: 1624 2330 2  
: 1625 2331 2  
: 1626 2332 2  
: 1627 2333 2  
: 1628 2334 2  
: 1629 2335 2  
: 1630 2336 2  
: 1631 2337 2  
: 1632 2338 2  
: 1633 2339 2  
: 1634 2340 2  
: 1635 2341 2  
: 1636 2342 2  
: 1637 2343 2  
: 1638 2344 2  
: 1639 2345 2  
: 1640 2346 2  
: 1641 2347 2  
: 1642 2348 2  
: 1643 2349 2  
: 1644 2350 2  
: 1645 2351 2  
: 1646 2352 2  
: 1647 2353 2  
: 1648 2354 2  
: 1649 2355 2  
: 1650 2356 2 TES:

FDL PARSE  
V04-000

## VAX-11 FDL Utilities SET\_CONNECT\_P

1651 2357 2 RETURN  
1652 2358 2  
1653 2359 1 END:

8 9  
16-Sep-1984 01:50:08 VAX-11 B1iss-32 v4.0-742 Page 52  
14-Sep-1984 12:31:19 DISK8VMSMASTER:[FDL.SRC]FDLPARSE.B32;1 (17)

000C 00000 SET_CONNECT P:									
0055	20	53	00000000G	00	9E	00002	.WORD	Save R2,R3	2244
0084		52	00000000G	00	9E	00009	MOVAB	FDL\$GL_NUMBER, R3	
0089		50	00000000G	00	D0	00010	MOVAB	FDL\$GL_SWITCH, R2	2284
00A5		23	00000000G	00	CF	00017	MOVL	FDL\$AB_PARSED_RAB, PARSED_RAB	
00BD	0050	0049	0042	0001F	1S:	.WORD	CASEL	FDL\$GL_SECONDARY, #35, #32	2288
00D7	0068	0061	005A	00027				2S-1S,-	
00F3	0070	0076	006F	0002F				3S-1S,-	
010F	009E	0097	0090	00037				4S-1S,-	
A0	00B6	00AF	00AA	0003F				5S-1S,-	
A0	00CB	00D0	00C4	00047				6S-1S,-	
A0	00EC	00E5	00DE	0004F				7S-1S,-	
A0	0108	0101	00FA	00057				8S-1S,-	
			0116	0005F				12S-1S,-	
								9S-1S,-	
								10S-1S,-	
								11S-1S,-	
								13S-1S,-	
								14S-1S,-	
								15S-1S,-	
								16S-1S,-	
								17S-1S,-	
								18S-1S,-	
								19S-1S,-	
								20S-1S,-	
								21S-1S,-	
								22S-1S,-	
								24S-1S,-	
								23S-1S,-	
								25S-1S,-	
								26S-1S,-	
								27S-1S,-	
								28S-1S,-	
								29S-1S,-	
								30S-1S,-	
								31S-1S,-	
								32S-1S,-	
								33S-1S,-	
								34S-1S,-	
A0	01	00	62	F0	00061	2S:	INSV	FDL\$GL_SWITCH, #0, #1, 4(PARSED_RAB)	2290
A0	01	03	62	F0	00068	3S:	INSV	FDL\$GL_SWITCH, #3, #1, 5(PARSED_RAB)	2292
	38	A0	63	D0	0006F	4S:	MOVL	FDL\$GL_NUMBER, 56(PARSED_RAB)	2294
	18	A0	63	D0	00074	5S:	MOVL	FDL\$GL_NUMBER, 24(PARSED_RAB)	2296
A0	01	00	62	F0	00079	6S:	INSV	FDL\$GL_SWITCH, #0, #1, 5(PARSED_RAB)	2298
			04	0007F			RET		

**FDLPARSE  
V04-000**      **VAX-11 FDL Utilities**  
**SET\_CONNECT\_P**

16-Sep-1984 01:50:08 VAX-11 Bliss-32 V4.0-742 Page 53  
14-Sep-1984 12:31:19 DISKS\$VMSMASTER:[FDL.SRC]FDLPARSE.B32;1 (17)

FD  
VO

05	A0	01	05	62	F0	00080	78:	INSV	FDL\$GL_SWITCH, #5, #1, 5(PARSED_RAB)	2300
04	A0	01	06	62	F0	00087	88:	RET	FDL\$GL_SWITCH, #6, #1, 4(PARSED_RAB)	2302
06	A0	01	05	62	F0	0008E	98:	INSV	FDL\$GL_SWITCH, #5, #1, 6(PARSED_RAB)	2304
06	A0	01	06	62	F0	00095	108:	INSV	FDL\$GL_SWITCH, #6, #1, 6(PARSED_RAB)	2306
05	A0	01	06	62	F0	0009C	118:	INSV	FDL\$GL_SWITCH, #6, #1, 5(PARSED_RAB)	2308
		35	A0	63	90	000A3	128:	MOV B	FDL\$GL_NUMBER, 53(PARSED_RAB)	2310
06	A0	01	00	62	F0	000AB	138:	RET	FDL\$GL_SWITCH, #0, #1, 6(PARSED_RAB)	2312
04	A0	01	02	62	F0	000AE	148:	INSV	FDL\$GL_SWITCH, #2, #1, 4(PARSED_RAB)	2314
06	A0	01	03	62	F0	000B5	158:	RET	FDL\$GL_SWITCH, #3, #1, 6(PARSED_RAB)	2316
06	A0	01	02	62	F0	000BD	168:	INSV	FDL\$GL_SWITCH, #2, #1, 6(PARSED_RAB)	2318
		37	A0	63	90	000C4	178:	RET	FDL\$GL_NUMBER, 55(PARSED_RAB)	2320
		36	A0	63	90	000C9	188:	MOV B	FDL\$GL_NUMBER, 54(PARSED_RAB)	2322
06	A0	01	04	62	F0	000CE	198:	RET	FDL\$GL_SWITCH, #4, #1, 6(PARSED_RAB)	2324
06	A0	01	07	62	F0	000D5	208:	INSV	FDL\$GL_SWITCH, #7, #1, 6(PARSED_RAB)	2326
05	A0	01	01	62	F0	000DC	218:	RET	FDL\$GL_SWITCH, #1, #1, 5(PARSED_RAB)	2328
04	A0	01	03	62	F0	000E3	228:	INSV	FDL\$GL_SWITCH, #3, #1, 4(PARSED_RAB)	2330
		1F	A0	63	90	000EA	238:	RET	FDL\$GL_NUMBER, 31(PARSED_RAB)	2332
07	A0	01	01	62	F0	000EF	248:	INSV	FDL\$GL_SWITCH, #1, #1, 7(PARSED_RAB)	2334
04	A0	01	01	62	F0	000F5	258:	RET	FDL\$GL_SWITCH, #1, #1, 4(PARSED_RAB)	2336
07	A0	01	07	62	F0	000FD	268:	INSV	FDL\$GL_SWITCH, #7, #1, 7(PARSED_RAB)	2338
07	A0	01	02	62	F0	00104	278:	RET	FDL\$GL_SWITCH, #2, #1, 7(PARSED_RAB)	2340
07	A0	01	06	62	F0	0010B	288:	INSV	FDL\$GL_SWITCH, #6, #1, 7(PARSED_RAB)	2342
07	A0	01	05	62	F0	00112	298:	INSV	FDL\$GL_SWITCH, #5, #1, 7(PARSED_RAB)	2344
07	A0	01	00	62	F0	00119	308:	RET	FDL\$GL_SWITCH, #0, #1, 7(PARSED_RAB)	2346
07	A0	01	03	62	F0	00120	318:	INSV	FDL\$GL_SWITCH, #3, #1, 7(PARSED_RAB)	2348
04	A0	01	04	62	F0	00127	328:	INSV	FDL\$GL_SWITCH, #4, #1, 4(PARSED_RAB)	2350
06	A0	01	01	62	F0	0012E	338:	RET	FDL\$GL_SWITCH, #1, #1, 6(PARSED_RAB)	2352
05	A0	01	02	62	F0	00135	348:	INSV	FDL\$GL_SWITCH, #2, #1, 5(PARSED_RAB)	2354
				62	F0	0013B	348:	RET		2359

FDLPARSE  
V04-000

VAX-11 FDL Utilities  
SET\_CONNECT\_P

: Routine Size: 316 bytes, Routine Base: \_FDLSCODE + 0A63

09  
16-Sep-1984 01:50:08  
14-Sep-1984 12:31:19

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[FDL.SRC]FDLPARSE.B32;1

Page 54 (17)

FD  
VO

```
1655 2360 1 XSBTTL 'SET PROT'  
1656 2361 1 ROUTINE SET_PROT : NOVALUE =  
1657 2362 1 **  
1658 2363 1 Functional Description:  
1659 2364 1 Fill in the blanks for the protection xab  
1660 2365 1 Calling Sequence:  
1661 2366 1 set_prot()  
1662 2367 1 Input Parameters:  
1663 2368 1 none  
1664 2369 1 Implicit Inputs:  
1665 2370 1 fdl$secondary - Secondary code  
1666 2371 1 Output Parameters:  
1667 2372 1 none  
1668 2373 1 Implicit Outputs:  
1669 2374 1 none  
1670 2375 1 Routine Value:  
1671 2376 1 none  
1672 2377 1 Routines Called:  
1673 2378 1 none  
1674 2379 1 Side Effects:  
1675 2380 1 none  
1676 2381 1 --  
1677 2382 1 BEGIN  
1678 2383 1 ! See if the protection xab has been allocated yet  
1679 2384 1  
1680 2385 1 IF .PROTECTION_XAB EQLU 0  
1681 2386 1 THEN  
1682 2387 1 ! Allocate the xab an enter it into the chain  
1683 2388 1 PROTECTION_XAB = ALLOCATE_XAB ( XABSC_PRO, 0 );  
1684 2389 1  
1685 2390 1 ! Set the fields according to the secondary  
1686 2391 1  
1687 2392 1  
1688 2393 1  
1689 2394 1  
1690 2395 1  
1691 2396 2  
1692 2397 2  
1693 2398 2  
1694 2399 2  
1695 2400 2  
1696 2401 2  
1697 2402 2  
1698 2403 2  
1699 2404 2  
1700 2405 2  
1701 2406 2  
1702 2407 2  
1703 2408 2  
1704 2409 2  
1705 2410 2  
1706 2411 2  
1707 2412 2  
1708 2413 2  
1709 2414 2  
1710 2415 2  
1711 2416 2  
SET [ FDLSC_MTPRO ] : PROTECTION_XAB [ XABSB_MTACC ] = .FDL$GL_QUALIFIER;  
[ FDLSC_PROT ] : PROTECTION_XAB [ XABSW_PRO ] = NOT .FDL$GL_PROTECTION;  
[ FDLSC_OWNER ] : PROTECTION_XAB [ XABSL_UIC ] = .FDL$GL_OWNER_UIC;  
TES;
```

```
: 1712 2417 2
: 1713 2418 2 RETURN
: 1714 2419 2
: 1715 2420 1 END:
```

0004 00000 SET_PROT:							
							2361
							2400
							2405
							2409
							2411
							2413
							2415
							2420

: Routine Size: 96 bytes,    Routine Base: \_FDL\$CODE + 0B9F

```
: 1717 2621 1 %$BTTL 'ALLOCATE_XAB'
: 1718 2622 1 ROUTINE ALLOCATE_XAB ( XAB_TYPE, XAB_NUM ) =
: 1719 2623 1 ++
: 1720 2624 1 Functional Description:
: 1721 2625 1     Allocates an RMS extended attribute block from virtual memory
: 1722 2626 1 *****
: 1723 2627 1     NOTE: THIS ROUTINE ASSUMES XABs ARE CONNECTED TO THE SFAB !!!
: 1724 2628 1     IT WILL NOT WORK WITH XABs THAT ARE CONNECTED TO THE SRAB !!!
: 1725 2629 1 *****
: 1726 2630 1
: 1727 2631 1 Calling Sequence:
: 1728 2632 1     allocate_xab( xab_type, xab_num )
: 1729 2633 1 Input Parameters:
: 1730 2634 1     xab_type      - The RMS code for the type of xab wanted ie. XABSC_xab
: 1731 2635 1     xab_num       - Which xab is desired (for key and area xabs)
: 1732 2636 1 Implicit Inputs:
: 1733 2637 1     none
: 1734 2638 1 Output Parameters:
: 1735 2639 1     none
: 1736 2640 1 Implicit Outputs:
: 1737 2641 1     none
: 1738 2642 1 Routine Value:
: 1739 2643 1     Pointer to the new xab (also pointed to by current xab)
: 1740 2644 1 Routines Called:
: 1741 2645 1     fdL$get_vM
: 1742 2646 1 Side Effects:
: 1743 2647 1     current_xab pointes to the new xab
: 1744 2648 1 --
: 1745 2649 1 BEGIN
: 1746 2650 1 LOCAL
: 1747 2651 1     XAB      : REF BLOCK [ ,BYTE ],
: 1748 2652 1     FOUND,
: 1749 2653 1     XAB_LEN,
: 1750 2654 1     NEW_XAB;
: 1751 2655 1
: 1752 2656 1     ! Find the size of the type of xab we want.
: 1753 2657 1
: 1754 2658 1
: 1755 2659 1
: 1756 2660 1
: 1757 2661 1
: 1758 2662 1
: 1759 2663 1
: 1760 2664 1
: 1761 2665 1
: 1762 2666 1
: 1763 2667 1
: 1764 2668 2
: 1765 2669 2
: 1766 2670 2
: 1767 2671 2
: 1768 2672 2
: 1769 2673 2
: 1770 2674 2
: 1771 2675 2
: 1772 2676 2
: 1773 2677 2
```

```
1774      2478 3      XAB_LEN = ( SELECTONEU .XAB_TYPE OF
1775      2479 3      SET
1776      2480 3      [ XABSC_ALL ] : XABSC_ALLLEN;
1777      2481 3      [ XABSC_DAT ] : XABSC_DATLEN;
1778      2482 3      [ XABSC_JNL ] : XABSC_JNLLLEN;
1779      2483 3      [ XABSC_KEY ] : XABSC_KEYLEN;
1780      2484 3      [ XABSC_PRO ] : XABSC_PROLEN;
1781      2485 3      [ XABSC_RDT ] : XABSC_RDTLEN;
1782      2486 3      TES );
1783      2487 3
1784      2488 3      FOUND = _CLEAR;
1785      2489 3
1786      2490 3      | See if the xab we need already exists
1787      2491 3      | (if we're in the second parse)
1788      2492 3
1789      2493 4      IF (
1790      2494 4      ( .FDLSAB_CTRL [ FDLSV_REPARSE ] )
1791      2495 4      AND
1792      2496 4      ( ( .XAB_TYPE EQLU XABSC_ALL ) OR ( .XAB_TYPE EQLU XABSC_KEY ) )
1793      2497 2      ) THEN
1794      2498 2      BEGIN
1795      2499 2
1796      2500 2      XAB = .FDLSAB_PARSED_FAB [ FABSL_XAB ];
1797      2501 2
1798      2502 3      WHILE .XAB NEQU 0
1799      2503 3      DO
1800      2504 4      BEGIN
1801      2505 4
1802      2506 5      IF (
1803      2507 7      ( ( .XAB_TYPE EQLU XABSC_ALL )
1804      2508 6      AND
1805      2509 7      ( .XAB [ XABSB_COD ] EQLU XABSC_ALL )
1806      2510 6      AND
1807      2511 6      ( .XAB [ XABSB_AID ] EQLU .XAB_NUM ))
1808      2512 5      OR
1809      2513 7      ( ( .XAB_TYPE EQLU XABSC_KEY )
1810      2514 6      AND
1811      2515 7      ( .XAB [ XABSB_COD ] EQLU XABSC_KEY )
1812      2516 6      AND
1813      2517 6      ( .XAB [ XABSB_REF ] EQLU .XAB_NUM ))
1814      2518 4      ) THEN
1815      2519 5      BEGIN
1816      2520 5
1817      2521 5      NEW_XAB = .XAB;
1818      2522 5      FOUND = SET;
1819      2523 5      EXITLOOP;
1820      2524 5
1821      2525 4      END;
1822      2526 4
1823      2527 4      XAB = .XAB [ XABSL_NXT ];
1824      2528 4
1825      2529 3
1826      2530 3
1827      2531 2
1828      2532 2
1829      2533 2
1830      2534 2      IF NOT .FOUND
1831      2535 2      THEN
```

```

: 1831      2535 3      BEGIN
: 1832      2536 3
: 1833      2537 3
: 1834      2538 3
: 1835      2539 3
: 1836      2540 3
: 1837      2541 3
: 1838      2542 3
: 1839      2543 3
: 1840      2544 3
: 1841      2545 3
: 1842      2546 3
: 1843      2547 3
: 1844      2548 3
: 1845      2549 3
: 1846      2550 3
: 1847      2551 3
: 1848      2552 3
: 1849      2553 2
: 1850      2554 2
: 1851      2555 2
: 1852      2556 2
: 1853      2557 2
: 1854      2558 2
: 1855      2559 2
: 1856      2560 2
: 1857      2561 2
: 1858      2562 2
: 1859      2563 2
: 1860      2564 2
: 1861      2565 2
: 1862      2566 2
: 1863      2567 2
: 1864      2568 2
: 1865      2569 2
: 1866      2570 2
: 1867      2571 2
: 1868      2572 1

      2      BEGIN
      3      ! Allocate a buffer for the new xab
      4      NEW_XAB = FDLSSGET_VMC(.XAB_LEN);
      5      ! If this is the first xab link it to the fab else just connect it to
      6      ! the last xab in the chain
      7      IF .FDL$AB_PARSED_FAB[FABSL_XAB] EQL 0
      8      THEN
      9          FDL$AB_PARSED_FAB[FABSL_XAB] = .NEW_XAB
     10      ELSE
     11          END_XAB[XABSL_NXT] = .NEW_XAB;
     12      END_XAB = .NEW_XAB;
     13      END;
     14
     15      ! Make this xab the current one
     16      CURRENT_XAB = .NEW_XAB;
     17
     18      IF NOT .FOUND
     19      THEN
     20          BEGIN
     21              ! Init. some stuff in it
     22              CURRENT_XAB[XABSB_COD] = .XAB_TYPE;
     23              CURRENT_XAB[XABSB_BLN] = .XAB_LEN;
     24              CURRENT_XAB[XABSL_NXT] = 0;
     25
     26          END;
     27
     28      RETURN .CURRENT_XAB
     29
     30      END;
  
```

## 007C 00000 ALLOCATE\_XAB:

56 00000000G	00 9E 00002	:WORD	Save R2,R3,R4,R5,R6	: 2422
55 00000000	00 9E 00009	MOVAB	FDL\$AB_PARSED_FAB, R6	
52 04	AC D0 00010	MOVAB	CURRENT_XAB, R5	
14	52 D1 00014	MOVL	XAB_TYPE, R2	: 2478
	05 12 00017	CMPL	R2, #20	: 2480
53	20 D0 00019	BNEQ	1\$	
	37 11 0001C	MOVL	#32, XAB_LEN	
12	52 D1 0001E	BRB	7\$	
	05 12 00021	CMPL	R2, #18	: 2481
53	20 D0 00023	BNEQ	2\$	
	5D 11 00026	MOVL	#44, XAB_LEN	
22	52 D1 00028	BRB	7\$	
	28: CMPL	R2, #34		: 2482

J 9  
16-Sep-1984 01:50:08 VAX-11 Bliss-32 v4.0-742 Page 60  
14-Sep-1984 12:31:19 DISK\$VMSMASTER:[FDL.SRC]FDLPARSE.B32:1 (19)

FDLPARSE  
V04-000

VAX-11 FDL Utilities  
ALLOCATE\_XAB

K<sup>9</sup>  
16-Sep-1984 01:50:08 VAX-11 BLISS-32 V4.0-742  
14-Sep-1984 12:31:19 DISK\$VM\$MASTER:[FDL.SRC]FDLPARSE.B32;1 Page 61  
(19)

01	A0	53	90	000D7	MOVB	XAB LEN, 1(R0)
50	06	A0	D4	000DB	CLRL	4(R0)
		65	D0	000DE	178:	MOVL CURRENT_XAB, R0
			04	000E1		RET

: 2565  
: 2566  
: 2570  
: 2572

; Routine Size: 226 bytes, Routine Base: \_FDL\$CODE + 0BFF

```
1870      2573 1 XSBTTL 'FIND_ID'
1871      2574 1 ROUTINE FIND_ID : NOVALUE =
1872      2575 1 ++
1873      2576 1
1874      2577 1 Functional Description:
1875      2578 1     Finds a file ID of a file specified by the FDL$STRING descriptor
1876      2579 1
1877      2580 1 Calling Sequence:
1878      2581 1     find_id()
1879      2582 1
1880      2583 1 Input Parameters:
1881      2584 1     none
1882      2585 1
1883      2586 1 Implicit Inputs:
1884      2587 1     none
1885      2588 1
1886      2589 1 Output Parameters:
1887      2590 1     none
1888      2591 1
1889      2592 1 Implicit Outputs:
1890      2593 1     none
1891      2594 1
1892      2595 1 Routine Value:
1893      2596 1     none
1894      2597 1
1895      2598 1 Routines Called:
1896      2599 1     fdls$get_vm
1897      2600 1
1898      2601 1 Side Effects:
1899      2602 1     none
1900      2603 1
1901      2604 1
1902      2605 1
1903      2606 1
1904      2607 1 --
1905      2608 1
1906      2609 2 BEGIN
1907      2610 2
1908      2611 2 LOCAL
1909      2612 2     FAB    : REF BLOCK [ ,BYTE ],
1910      2613 2     NAM    : REF BLOCK [ ,BYTE ];
1911      2614 2
1912      2615 2     ! Get the address space for the FAB and the Name block
1913      2616 2
1914      2617 2     FAB = FDLSGET_VMC( FAB$K_BLN );
1915      2618 2
1916      2619 2     NAM = FDLSGET_VMC( NAM$K_BLN + ESA_BUF_SIZ );
1917      2620 2
1918      2621 2     +-----+
1919      2622 2     |   nam blk   |
1920      2623 2     +-----+
1921      2624 2     |   exp str buf   |
1922      2625 2     +-----+
1923      2626 2
1924      2627 2     ! Init the blocks and fill in all of the good stuff
1925      2628 2
1926      P 2629 2     $FAB_INIT ( FAB = .FAB,
```

```

1927 P 2630 2      FNA = .FDLSAB_STRING [ DSCSA_POINTER ],
1928 P 2631 2      FNS = .FDLSAB_STRING [ DSCSW_LENGTH ],
1929 P 2632 2      NAM = .NAM );
1930 P 2633 2
1931 P 2634 2      $NAM_INIT ( ESA = .NAM + NAMSK_BLN,
1932 P 2635 2      ESS = ESA BUF_SIZ,
1933 P 2636 2      NAM = .NAM );
1934 P 2637 2
1935 P 2638 2      ! Parse and search4 for the file
1936 P 2639 2
1937 P 2640 3      IF SPARSE( FAB=.FAB )
1938 P 2641 2      THEN
1939 P 2642 2
1940 P 2643 3      IF $SEARCH( FAB=.FAB )
1941 P 2644 2      THEN
1942 P 2645 3      BEGIN
1943 P 2646 3
1944 P 2647 3      ! Get the old file ID
1945 P 2648 3
1946 P 2649 3      FDLSGL_FID1 = .NAM [ NAMSW_FID_NUM ];
1947 P 2650 3      FDLSGL_FID2 = .NAM [ NAMSW_FID_SEQ ];
1948 P 2651 3      FDLSGL_FID3 = .NAM [ NAMSW_FID_RVN ];
1949 P 2652 3
1950 P 2653 3
1951 P 2654 2      END
1952 P 2655 2      ELSE SIGNAL( FDLS_RFLOC )
1953 P 2656 2
1954 P 2657 2      ELSE SIGNAL( FDLS_RFLOC );
1955 P 2658 2
1956 P 2659 2
1957 P 2660 2      ! Deallocate the space we used
1958 P 2661 2      FDLSFREE_VM( FABSK_BLN, .FAB );
1959 P 2662 2      FDLSFREE_VM( NAMSK_BLN+ESA_BUF_SIZ, .NAM );
1960 P 2663 2
1961 P 2664 2      RETURN
1962 P 2665 2
1963 P 2666 1      END;

```

## .EXTRN SY\$PARSE, SY\$SEARCH

				03FC 00000 FIND_ID: .WORD	Save R2,R3,R4,R5,R6,R7,R8,R9	2574
			59 00000000V	00 9E 00002	MOVAB FDLSSGET VM, R9	
			58 00000000V	00 9E 00009	MOVAB FDLSSFREE VM, R8	
			7E 50	8F 9A 00010	MOVZBL #80, -(SP)	2617
			69	01 FB 00014	CALLS #1, FDLSSGET_VM	
			57	50 D0 00017	MOVL R0, FAB	
			7E 015F	8F 3C 0001A	MOVZWL #351, -(SP)	2619
			69	01 FB 0001F	CALLS #1, FDLSSGET_VM	
			56	50 D0 00022	MOVL R0, NAM	
0050	8F	00	6E	00 2C 00025	MOVCS #0, (SP), #0, #80, (FAB)	2632
				67 0002C		
			16 A7	5003 8F B0 0002D	MOVW #20483, (FAB)	
			1F A7	02 90 00032	MOVB #2, 22(FAB)	
			28 A7	02 90 00036	MOVB #2, 31(FAB)	
				56 D0 0003A	MOVL NAM, 40(FAB)	

0060	8F	00	2C 34	A7 00000000G	00 6E	00 90	0003E 00046	MOVL MOVBL MOVCS	FDL\$AB_STRING+4, 44(FAB) FDL\$AB_STRING, 52(FAB) #0, (SP), #0, #96, (NAM)	2636
				66 6002	8F 01	B0 8E	00056 0005B	MOVW MNEG.B	#24578, (NAM) #1, 10(NAM)	
			0A 0C	A6 A6	60	A6 57	9E 0005F DD 00064	MOVAB PUSHL	96(R6), 12(NAM) FAB	2640
				00000000G 00	01	FB 50	00066 E9 0006D	CALLS BLBC	#1, SY\$PARSE R0, 1\$	
				26		57	DD 00070	PUSHL	FAB	2643
				00000000G 00	01	FB 50	00072 E9 00079	CALLS BLBC	#1, SY\$SEARCH R0, 1\$	
				1A	24	A6 50	3C 0007C	MOVZWL	36(NAM), FDL\$GL_FID1	2649
				00000000G 00	26	A6 50	3C 00084	MOVZWL	38(NAM), FDL\$GL_FID2	2650
				00000000G 00	28	A6 0D	3C 0008C 11 00094	MOVZWL BRB	40(NAM), FDL\$GL_FID3 2\$	2651
						8F 01	DD 00096 FB 0009C	PUSHL CALLS	#FDL\$RFLOC #1, LIB\$SIGNAL	2657
				00000000G 00		57	DD 000A3	PUSHL	FAB	2661
						8F 02	9A 000A5	MOVZBL	#80, -(SP)	
						68	FB 000A9	CALLS	#2, FDL\$\$FREE_VM	
						7E 015F	56 02	PUSHL	NAM	2662
						68	3C 000AE	MOVZWL	#351, -(SP)	
							FB 000B3	CALLS	#2, FDL\$\$FREE_VM	
							04 000B6	RET		2666

: Routine Size: 183 bytes, Routine Base: \_FDL\$CODE + 0CE1

```
1965 2667 1 XSBTTL 'GET_VM'  
1966 2668 1 GLOBAL ROUTINE FDLS$GET_VM( BYTES ) =  
1967 2669 1 ++  
1968 2670 1  
1969 2671 1 Functional Description:  
1970 2672 1 Allocate virtual memory and zeros it  
1971 2673 1 Calling Sequence:  
1972 2674 1 fdls$get_vm( bytes )  
1973 2675 1 Input Parameters:  
1974 2676 1 bytes - number of bytes to allocate  
1975 2677 1 Implicit Inputs:  
1976 2678 1 none  
1977 2679 1 Output Parameters:  
1978 2680 1 none  
1979 2681 1 Implicit Outputs:  
1980 2682 1 none  
1981 2683 1 Routine Value:  
1982 2684 1 2685 1 address of the start of the buffer  
1983 2686 1  
1984 2687 1 Routine Called:  
1985 2688 1 lib$get_vm  
1986 2689 1  
1987 2690 1 Side Effects:  
1988 2691 1 none  
1989 2692 1  
1990 2693 1  
1991 2694 1  
1992 2695 1  
1993 2696 1  
1994 2697 1  
1995 2698 1  
1996 2699 1  
1997 2700 1  
1998 2701 1  
1999 2702 1  
2000 2703 1  
2001 2704 1  
2002 2705 2  
2003 2706 2  
2004 2707 2  
2005 2708 2  
2006 2709 2  
2007 2710 2  
2008 2711 2  
2009 2712 2  
2010 2713 2  
2011 2714 2  
2012 2715 2  
2013 2716 2  
2014 2717 2  
2015 2718 2  
2016 2719 2  
2017 2720 2  
2018 2721 2  
2019 2722 1  
2020 2722 1  
--  
BEGIN  
LOCAL  
    VM_POINTER;  
    ! If we don't succeed signal an error and stop  
    IF NOT LIBSGET_VM( BYTES,VM_POINTER )  
    THEN  
        SIGNAL_STOP( FDLS_INSVIRMEM );  
    ! Zero this address space  
    CH$FILL( 0,,BYTES,,VM_POINTER );  
    RETURN .VM_POINTER  
END;
```

		5E	003C 00000	.ENTRY    FDLSS\$GET_VM, Save R2,R3,R4,R5	: 2668
		04	04 C2 00002	SUBL2    #4, SP	
		00	5E DD 00005	PUSHL    SP	: 2712
	00000000G	00	AC 9F 00007	PUSHAB    BYTES	
		0D	02 FB 0000A	CALLS    #2, LIB\$GET_VM	
		00	50 E8 00011	BLBS    R0, 1\$	
	00000000G	00	8F DD 00014	PUSHL    #FDLS_INSVIRMEM	: 2714
04 AC	00	00	01 FB 0001A	CALLS    #1, LIB\$STOP	
		6E	00 2C 00021	MOVCS    #0, (SP), #0, BYTES, AVM_POINTER	: 2718
		00	BE 00027		
		50	6E D0 00029	MOVL    VM_POINTER, R0	: 2720
			04 0002C	RET	: 2722

; Routine Size: 45 bytes,    Routine Base: \_FDL\$CODE + 0D98

```
: 2022    2723 1 %SBTTL 'FREE_VM'
: 2023    2724 1 GLOBAL ROUTINE FDL$$FREE_VM( BYTES,ADDR ) : NOVALUE =
: 2024    2725 1 ++
: 2025    2726 1
: 2026    2727 1 Functional Description:
: 2027    2728 1     Deallocate virtual memory
: 2028    2729 1
: 2029    2730 1 Calling Sequence:
: 2030    2731 1     fdl$$free_vm( bytes,addr )
: 2031    2732 1
: 2032    2733 1 Input Parameters:
: 2033    2734 1     bytes - number of bytes to deallocate
: 2034    2735 1     addr  - address of block
: 2035    2736 1
: 2036    2737 1 Implicit Inputs:
: 2037    2738 1     none
: 2038    2739 1
: 2039    2740 1 Output Parameters:
: 2040    2741 1     none
: 2041    2742 1
: 2042    2743 1 Implicit Outputs:
: 2043    2744 1     none
: 2044    2745 1
: 2045    2746 1 Routine Value:
: 2046    2747 1     none
: 2047    2748 1
: 2048    2749 1 Routine Called:
: 2049    2750 1     lib$free_vm
: 2050    2751 1
: 2051    2752 1 Side Effects:
: 2052    2753 1     none
: 2053    2754 1
: 2054    2755 1
: 2055    2756 1
: 2056    2757 1
: 2057    2758 1
: 2058    2759 1 --+
: 2059    2760 1
: 2060    2761 2 BEGIN
: 2061    2762 2
: 2062    2763 2 LOCAL
: 2063    2764 2     STATUS;
: 2064    2765 2
: 2065    2766 2     ! If we don't succeed signal an error and stop
: 2066    2767 2
: 2067    2768 3 IF NOT ( STATUS = LIB$FREE_VM ( BYTES,ADDR ) )
: 2068    2769 2 THEN
: 2069    2770 2     SIGNAL_STOP ( .STATUS );
: 2070    2771 2
: 2071    2772 2 RETURN
: 2072    2773 2
: 2073    2774 1 END;
```

FDLPARSE  
V04-000

VAX-11 FDL Utilities  
FREE\_VM

E 10

16-Sep-1984 01:50:08  
14-Sep-1984 12:31:19

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[FDL.SRC]FDLPARSE.B32;1 (22)

Page 68  
FD  
VO

00000000G 00	08 0000 0000	.ENTRY FDL\$FREE_VM, Save nothing	: 2724
09	04 AC 9F 00002	PUSHAB ADDR	: 2768
	02 FB 00008	PUSHAB BYTES	:
	50 EB 0000F	CALLS #2, LIB\$FREE_VM	:
00000000G 00	50 DD 00012	BLBS STATUS, 1\$	:
	01 FB 00014	PUSHL STATUS	: 2770
	04 0001B 1\$:	CALLS #1, LIB\$STOP	:
		RET	: 2774

; Routine Size: 28 bytes, Routine Base: \_FDL\$CODE + 0DC5

: 2074 2775 1  
: 2075 2776 0 END ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

#### PSECT SUMMARY

Name	Bytes	Attributes
_FDL\$OWN	28 NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON, PIC,ALIGN(2)	
_FDL\$CODE	3553 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)	

#### Library Statistics

File	----- Symbols -----			Pages Mapped	Processing Time
	Total	Loaded	Percent		
\$_\$255\$DUA2B:[SYSLIB]STARLET.L32;1	9776	244	2	581	00:01.0

#### COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:FDLPARSE/OBJ=OBJ\$:FDLPARSE MSRC\$:FDLPARSE/UPDATE=(ENH\$:FDLPARSE)

Size: 3553 code + 28 data bytes  
Run Time: 00:59.3  
Elapsed Time: 03:08.7  
Lines/CPU Min: 2809  
Lexemes/CPU-Min: 21493  
Memory Used: 276 pages  
Compilation Complete

0177 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

FOLPARSE  
LIS

FOLPARDEF  
LIS

FOLSO MSG  
LIS

FOLTABLES  
LIS

FOLGEN AB

FOLMSG  
LIS